ILLINOIS UNIVERSITY DEPARTMENT
OF COMPUTER SCIENCE

REPORT

[redacted]

UIUCDCS-R-76-841

NOR NETWORK TRANSDUCTION PROCEDURES BASED ON CONNECTABLE AND DISCONNECTABLE CONDITIONS
(Principles of NOR Network Transduction Programs NETTRA-G1 and NETTRA-G2)

by

Y. Kambayashi and J. N. Culliney

December, 1976

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-76-841

NOR NETWORK TRANSDUCTION PROCEDURES BASED ON CONNECTABLE AND DISCONNECTABLE CONDITIONS

(Principles of NOR Network Transduction Programs NETTRA-G1 and NETTRA-G2)

by

Y. Kambayashi
J. N. Culliney

December, 1976

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

This work was supported in part by the National Science Foundation under Grant No. DCR73-03421.

ACKNOWLEDGMENT

The authors are grateful to Professor S. Muroga for his guidance and careful reading of this report. The authors would like to thank Dr. H.C. Lai for his valuable discussions throughout the period of this research, and Mrs. Ruby Taylor for typing it.

TABLE OF CONTENTS

1. INTRODUCTION

Based on the concept of compatible sets of permissible functions (CSPF's) discussed in a previous paper, [5], network transduction (transformation and reduction) procedures have been developed to derive simplified networks from given feed-forward multiple-output networks of NOR gates. In this paper, one of the major categories of transduction procedures is discussed. Unlike the pruning procedures discussed in [3] which are unable to create new connections, transduction procedures of this category are able to make new connections among gates (or external variables).

First, conditions for the connectability and disconnectability of a connection from a NOR gate (or external variable) to another NOR gate are established. Based on these conditions, a basic transduction procedure is proposed which essentially deals with only a single gate of a given network (although others may be affected as a consequence). From this, a more sophisticated transduction procedure is developed by adapting the basic procedure for efficient repetitive application. This procedure attempts to enable the removal of connections and gates throughout the network. A third transduction procedure is created from the second by causing it to focus upon the removal of a specific gate.

In the transductions, various orderings of gates are employed which define their priorities during the performance of various operations. Since these orderings affect the efficiency of the procedures, the choice of appropriate orderings is discussed.

Computer program implementations, designated NETTRA-G1 and NETTRA-G2, were written in FORTRAN IV for the two transduction procedures other than the basic procedure. (A reference manual, [2], is available for users of the programs.) Some modifications are discussed which were made of the original

procedures in the interest of computational efficiency. Then, computational results by these programs are presented.

Modifications of the two programs have been made which allow them to perform network transduction under fan-in, fan-out, and level restrictions. The corresponding alterations of the programs will be discussed elsewhere.

The notations and basic concepts defined in [5] will be used throughout this paper.

Let W be the total number of connections except connections from output terminals. The cost of a network is represented by

$$AR + BW.$$

Here, A and B are cost coefficients. In this paper A = 100 and B = 1 are chosen. Therefore, the reduction of the number of NOR gates is more important than the reduction of the number of connections.

2. BASIC PROCEDURE

In this section, a basic network transduction procedure based on connectable and disconnectable conditions is developed. More advanced transduction procedures are discussed in succeeding sections.

Definition 2.1 A gate or an input terminal†, v_i, is said to be connectable to a gate v_j with respect to a set of permissible functions of v_j, $G(v_j)$, if (1) the output function of v_j remains in $G(v_j)$ after adding connection c_{ij}, and (2) the network obtained after this input addition is loop-free.

Definition 2.2 Connection c_{ij} is said to be disconnectable from a gate v_j with respect to a set of permissible functions of v_j, $G(v_j)$, if the output function of v_j remains in $G(v_j)$ after removing c_{ij} from the network.

The following two theorems are bases of the transduction procedures to be developed. The proofs are obvious.

Theorem 2.1 [5]‡ A gate or an input terminal, v_i, is connectable to a gate, v_j, with respect to set $G(v_j)$ if and only if the following conditions are satisfied:

(1) For all d such that $f^{(d)}(v_i) = 1$,

$$G^{(d)}(v_j) = 0 \text{ or } *.$$

where $f^{(d)}(v_i)$ is the d-th component of the output vector of gate v_i.

---

†The concept of "input terminals" was introduced in [5] to simplify notational expressions in the many cases where equivalent treatment of gates and external variables is necessary. Since in this paper only non-complemented variables are assumed to be available to the network, the terms "input terminals" and "external variables" can be considered interchangeable.

‡Theorem 5.3 in [5].

(2) v_i is not contained in $S(v_j)$ where $S(v_j)$ denotes the set of successor gates of gate v_j.

Theorem 2.2 [5]† Connection c_{ij} is disconnectable from gate v_j with respect to set $G(v_j)$ if and only if the following condition is satisfied: For all d such that $f^{(d)}(v_i) = 1$, either:

$$G^{(d)}(v_j) = * \text{ or}$$

$$\bigvee_{\substack{v \in IP(v_j) \\ v \neq v_i}} f^{(d)}(v) = 1$$

where $IP(v_j)$ denotes the set of all immediate predecessor gates of gate v_j.

(Note that in both theorems, v_i may be an external variable. In such a case, $f(v)^{(d)}$ is $x^{(d)}$.)

Definition 2.3 $K(v_j)$, the set of connectable functions to gate v_j with respect to $G(v_j)$ is the set of all functions such that the function realized by v_j remains in $G(v_j)$ after adding any one of the functions as an input of v_j.

Lemma 2.1 The set $K(v_j)$ of all connectable functions to a gate v_j with respect to $G(v_j)$ is given, in vector representation, by:

$K^{(d)}(v_j) = 0$ for all d such that $G^{(d)}(v_j) = 1$, and

$K^{(d)}(v_j) = *$ for all other d's.

Lemma 2.2 If the functions in any subset of $K(v_j)$ are added simultaneously as inputs to gate v_i, the resulting function realized by v_j remains in $G(v_j)$.

Clearly, if $f(v_i)$ is in $K(v_j)$ and $v_i \notin S(v_j)$, v_i is connectable to

†Lemma 4.1 in [5].

v_j with respect to $G(v_j)$.

Definition 2.4 Let $Q(v_j)$ denote the set of all input terminals and gates which are connectable to v_j with respect to $G(v_j)$. Input terminal or gate v_i in $Q(v_j)$† is said to be an essential input of v_j with respect to $G(v_j)$ if gate v_j with input connections from all elements in $\{Q(v_j) - \{v_j\}\}$ realizes a function not contained in $G(v_j)$. Let $Q_o(v_j)$ denote the set of all input terminals and gates which are essential inputs of v_j with respect to $G(v_j)$. A v_i in $Q(v_j)$ which is not an essential input with respect to $G(v_j)$ (i.e., not in $Q_o(v_j)$) is called a non-essential input with respect to $G(v_j)$.

If v_i is an essential input of v_j with respect to $G(v_j)$, a connection from v_i to v_j is necessary to maintain an output function of v_i within $G(v_j)$.

Lemma 2.3 Input terminal or gate v_i in $Q(v_j)$ is an essential input of v_j with respect to $G(v_j)$ if and only if there exists at least one d such that:

$$G^{(d)}(v_j) = 0, \quad f^{(d)}(v_i) = 1, \text{ and } \bigvee_{\substack{v \in Q(v_j) \\ v \neq v_i}} f^{(d)}(v) = 0.$$

Definition 2.5 The component $f^{(d)}(v_j)(=1)$ for each d for which the condition in Lemma 2.3 holds is said to be an essential 1, with respect to $G(v_j)$, of the essential input v_i of v_j.

Note that an input terminal or gate v_i is an essential input to a gate v_j with respect to $G(v_j)$ if and only if the vector representation of $f(v_i)$ contains essential 1's with respect to $G(v_j)$.‡

---

†Note that while $K(v_j)$ has a vector representation, set $Q(v_j)$ does not.

‡The qualifying phrase "with respect to $G(v_j)$" may sometimes be omitted when it is clear from the context of the discussion.

Definition 2.6 If there exists an input v_i of a gate v_j and a d such that:

$$G^{(d)}(v_j) = 0 \text{ and } f^{(d)}(v_i) = 1,$$

then v_i is said to cover $G^{(d)}(v_j)$, either or both of v_i and $f^{(d)}(v_i)$ may be referred to as a cover of $G^{(d)}(v_j)$, and $G^{(d)}$ is said to be covered.

A basic network transduction procedure based on the preceding concepts is as follows:

Procedure 2.1 [Procedure RI] (a procedure to Replace Inputs of a particular gate v_j.)

(1) Select a particular gate v_j in a given network.

(2) Calculate a set of permissible functions, $G(v_j)$, for v_j. (A compatible set of permissible functions, $G_c(v_j)$ may be used.)

(3) Calculate a set $K(v_j)$ of connectable functions with respect to $G(v_j)$.

(4) Calculate the set $Q(v_j)$ of connectable input terminals and gates with respect to $G(v_j)$.

(5) Calculate the subset $Q_o(v_j)$ of all essential inputs in $Q(v_j)$.

(6) Select a subset, Q', of $Q(v_j)$ satisfying:

$$Q_o(v_j) \subseteq Q' \subseteq Q(v_j),$$

$$\bigwedge_{v \in Q'} \overline{f}(v) \in G(v_j).$$

(7) Let the new set of inputs for v_j be Q', resulting in a modified network. Calculate CSPF's for this network, individually removing (redundant) connections satisfying the conditions for disconnectability (see Theorem 2.2).

(8) If the network obtained after Step (7) is of less cost than the original, or if all possible subsets Q' have been exhausted in Step (6), terminate the procedure. Otherwise, restore the original network, return to Step

(6), and select a new Q' not previously chosen.

The following example demonstrates the use of Procedure RI.

Example 2.1 Fig. 2.1 shows a network which realizes the function

$$f = \bar{x}_1 x_2 \vee \bar{x}_1 \bar{x}_3 \vee x_1 \bar{x}_2 x_3.$$

Functions realized at the input terminals and gates are as follows:

$$f(v_1) = x_1 = (0\ 0\ 0\ 0\quad 1\ 1\ 1\ 1)$$
$$f(v_2) = x_2 = (0\ 0\ 1\ 1\quad 0\ 0\ 1\ 1)$$
$$f(v_3) = x_3 = (0\ 1\ 0\ 1\quad 0\ 1\ 0\ 1)$$
$$f(v_4) = (1\ 0\ 1\ 1\quad 0\ 1\ 0\ 0)$$
$$f(v_5) = (0\ 0\ 0\ 0\quad 0\ 0\ 1\ 1)$$
$$f(v_6) = (0\ 0\ 0\ 0\quad 1\ 0\ 1\ 0)$$
$$f(v_7) = (0\ 1\ 0\ 0\quad 0\ 0\ 0\ 0)$$
$$f(v_8) = (1\ 1\ 0\ 0\quad 1\ 1\ 0\ 0)$$
$$f(v_9) = (1\ 1\ 1\ 1\quad 0\ 0\ 0\ 0)$$
$$f(v_{10}) = (1\ 0\ 1\ 0\quad 1\ 0\ 1\ 0)$$

Fig. 2.2.(a) shows the same network; the illustration, however, is simplified due to the omission of input terminals.†

(1) Gate v_8 is selected.

(2) $G_c(v_8)$ is calculated as follows (see [5] for general procedure).

First, $G_c(v_5)$ is calculated from $G_c(v_4)$.

$$G_c(v_4) = f(v_4) = (1\ 0\ 1\ 1\quad 0\ 1\ 0\ 0),$$
$$f(v_6) \vee f(v_7) = (0\ 1\ 0\ 0\quad 1\ 0\ 1\ 0),$$
$$f(v_5) = (0\ 0\ 0\ 0\quad 0\ 0\ 1\ 1),$$
$$G_c(v_5) = (0\ *\ 0\ 0\quad *\ 0\ *\ 1).$$

For all d such that $G_c^{(d)}(v_4) = 1$, $G_c^{(d)}(v_5) = 0$. For all d such that

†Input terminals are omitted in all succeeding network illustrations for simplicity.

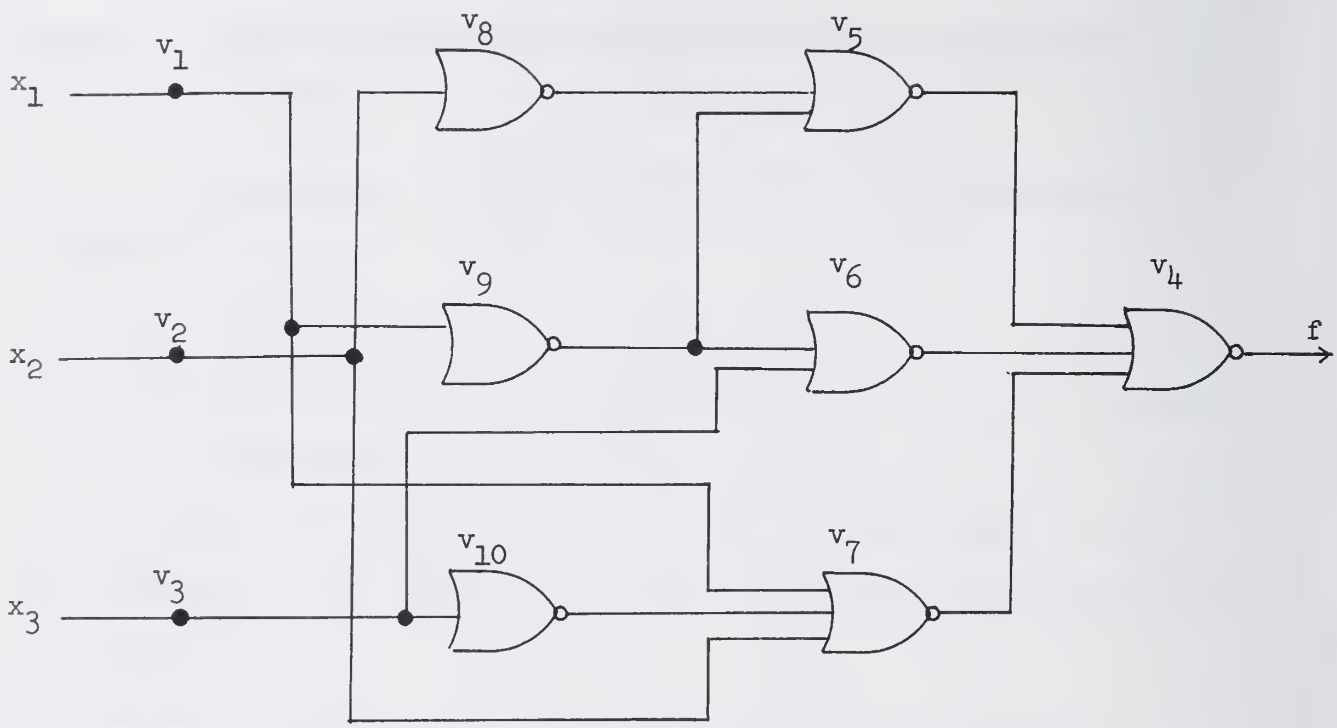


Fig. 2.1 Given network for f in Example 2.1.

$G_c^{(d)}(v_4) = 1$, $G_c^{(d)}(v_5) = 0$. For all d such that $G_c^{(d)}(v_4) = 0$ and $f^{(d)}(v_6) \vee f^{(d)}(v_7) = 0$, $G_c^{(d)}(v_5) = 1$ (necessary to assume $f^{(d)}(v_4) = 0$). For all other d, $G_c^{(d)}(v_5) = *$. Similarly, $G_c(v_8)$ is calculated using $G_c(v_5)$, $f(v_9)$, and $f(v_8)$.

$$G_c(v_8) = (* * * * \;\; * 1 * 0).$$

(3) The set, $K(v_8)$, of connectable functions to v_8 is simply determined.

$$K(v_8) = (* * * * \;\; * 0 * *).$$

(4) The following v_i are found to satisfy $f^{(6)}(v_i) = 0$:

$$v_2, v_5, v_6, v_7, v_9, \text{ and } v_{10}.$$

Since all except v_5 satisfy $v_i \notin S(v_8)$,

$$Q(v_8) = \{v_2, v_6, v_7, v_9, v_{10}\}.$$

(5) Of the elements of $Q(v_8)$, only v_2 is an essential input to v_8 with respect to $G_c(v_8)$ since

$$G_c^{(8)}(v_8) = 0, \quad f^{(8)}(v_2) = 1, \text{ and } \bigvee_{\substack{v \in Q(v_8) \\ v \neq v_2}} f^{(8)}(v) = 0.$$

Thus

$$Q_o(v_j) = \{v_2\}.$$

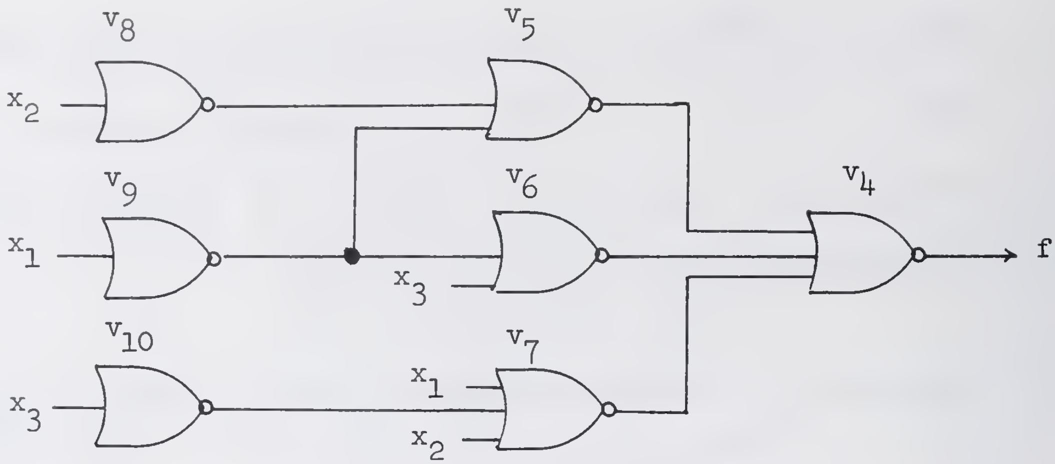
(6) Let Q' be $\{v_2, v_6\}$. A new connection is added from v_6 to v_8 (see Fig. 2.2(b)).

(7) During the calculation of CSPF's, the connection from v_6 to v_4 is removed. Since

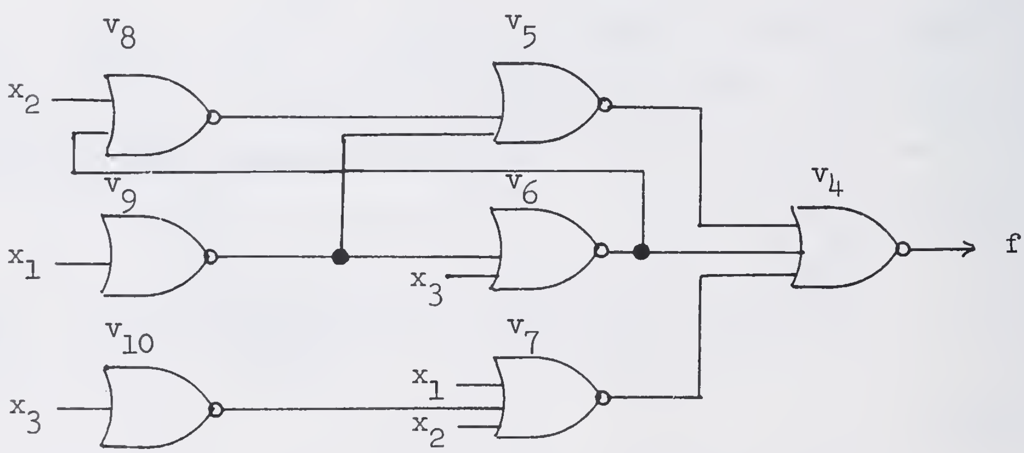
$$G_c(v_4) = f(v_4) = (1\ 0\ 1\ 1 \quad 0\ 1\ 0\ 0),$$
$$f(v_5) = (0\ 0\ 0\ 0 \quad 1\ 0\ 1\ 1),$$
$$f(v_6) = (0\ 0\ 0\ 0 \quad 1\ 0\ 1\ 0),$$
$$f(v_7) = (0\ 1\ 0\ 0 \quad 0\ 0\ 0\ 0),$$

connection $c_{6,4}$ clearly satisfies the condition for disconnectability (Theorem 2.2) and can be removed as shown in Fig. 2.2(c).

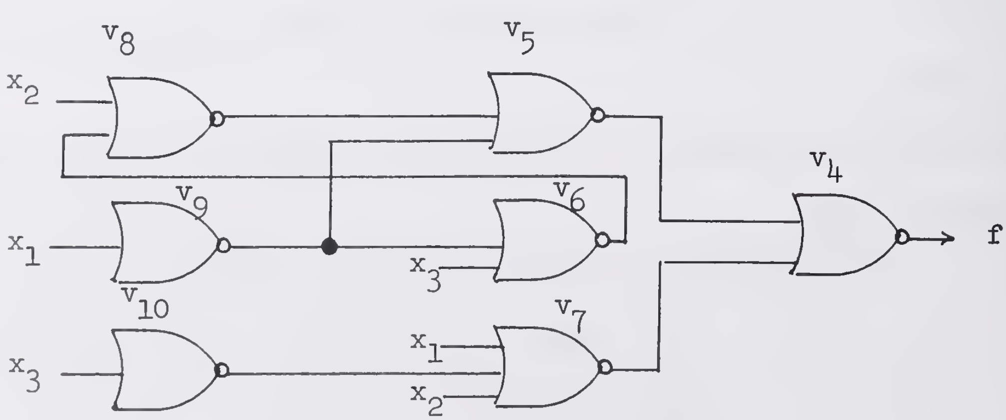
Continuing the calculation of CSPF's, connection $c_{9,6}$ is also found to



(a) Initial network.

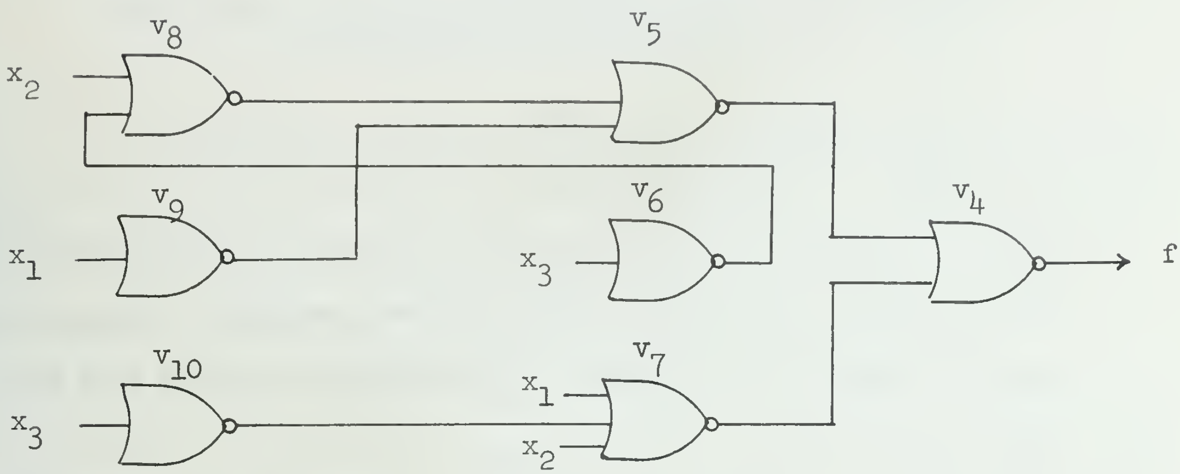


(b) Network after choice of $Q' = \{v_2, v_6\}$ for v_8.

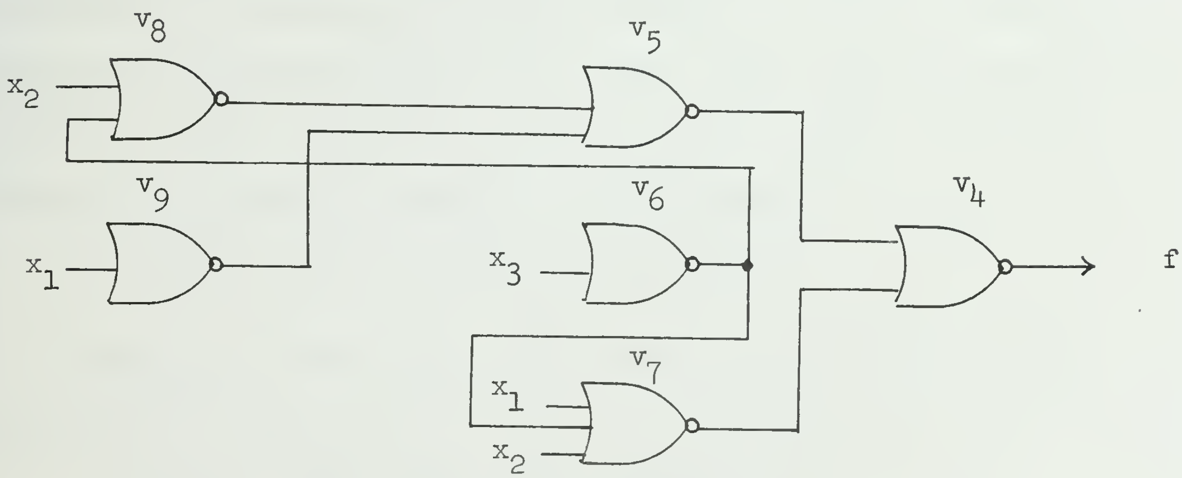


(c) Network after removal of $c_{6,4}$.

Fig. 2.2 Example for Procedure RI.



(d) Network after removal of $c_{9,6}$.



(e) Final network, v_{10} removed.

Fig. 2.2 (cont.) Example for Procedure RI.

be removable after determining $G_c(v_6)$:

$$G_c(v_6) = (* * * * \ 1 0 * *),$$

$$f(v_3) = (0 1 0 1 \ 0 1 0 1),$$

$$f(v_9) = (1 1 1 1 \ 0 0 0 0).$$

After the removal of $c_{9,6}$, the outputs of v_6 and v_{10} in the resulting network (Fig.2.2 (d)) have obviously become identical. Consequently, V_{10} is replaced by v_6 to produce a final network with one less gate (see Fig. 2.2(e)).

(8) The cost of the original network was 713. Since the cost of the final network has been reduced to 611, the procedure terminates.

This same result could be achieved by choosing $Q' = \{v_2, v_{10}\}$ in Step 6.

3. NOR NETWORK TRANSDUCTION PROCEDURE BASED ON CONNECTABLE AND DISCONNECTABLE CONDITIONS

In this section, an efficient network transduction procedure is developed based on Procedure RI of the previous section. In Procedure RI there are many possible choices in Step (1) (selection of the particular gate for the procedure) and Step (6) (selection of a new input set for the gate). In order to find the best possible result obtainable by a series of applications of the procedure, an impractical number of combinations of these choices must be exhausted. For this reason, an efficient method for the repetitive application of the procedure is desirable. In this section, such a procedure using a preference ordering is given. This procedure allows connections and disconnections of connections to be made throughout the network without the necessity of recalculating the CSPF's each time. Also, upon termination of the procedure, CSPF's are known for all gates, enabling other transduction procedures to be applied for possible further reduction.

An example is given to explain the procedure.

<u>Example 3.1</u> Fig. 3.1(a) shows a network which realizes the following 4-variable function f:

$$f = x_1x_3x_4 \vee x_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3 \vee \bar{x}_1\bar{x}_4 \vee \bar{x}_3\bar{x}_4.$$

This network is the first solution found during a logical design by a Branch-and-Bound method [9], [10] for function f.

Let an ordering r be defined as follows:

$$r(v_i) = i-4 \text{ for } i,\ 5 \leq i \leq 14 \text{ (gates)}$$

$$r(v_i) = i+10 \text{ for } i,\ 1 \leq i \leq 4 \text{ (input terminals)}.$$

When an input set for a gate is determined, the use of v_i with larger $r(v_i)$ is preferred (thus, input terminals are most preferred).

The functions realized at gates and input terminals are as follows:

$f(v_1) = x_1 = (0\ 0\ 0\ 0\ \ 0\ 0\ 0\ 0\ \ 1\ 1\ 1\ 1\ \ 1\ 1\ 1\ 1)$

$f(v_2) = x_2 = (0\ 0\ 0\ 0\ \ 1\ 1\ 1\ 1\ \ 0\ 0\ 0\ 0\ \ 1\ 1\ 1\ 1)$

$f(v_3) = x_3 = (0\ 0\ 1\ 1\ \ 0\ 0\ 1\ 1\ \ 0\ 0\ 1\ 1\ \ 0\ 0\ 1\ 1)$

$f(v_4) = x_4 = (0\ 1\ 0\ 1\ \ 0\ 1\ 0\ 1\ \ 0\ 1\ 0\ 1\ \ 0\ 1\ 0\ 1)$

$f(v_5) = (1\ 1\ 1\ 0\ \ 1\ 1\ 1\ 0\ \ 1\ 1\ 1\ 1\ \ 1\ 0\ 0\ 1)$

$f(v_6) = (0\ 0\ 0\ 0\ \ 0\ 0\ 0\ 0\ \ 0\ 0\ 0\ 0\ \ 0\ 0\ 1\ 0)$

$f(v_7) = (0\ 0\ 0\ 0\ \ 0\ 0\ 0\ 0\ \ 0\ 0\ 0\ 0\ \ 0\ 1\ 0\ 0)$

$f(v_8) = (0\ 0\ 0\ 1\ \ 0\ 0\ 0\ 1\ \ 0\ 0\ 0\ 0\ \ 0\ 0\ 0\ 0)$

$f(v_9) = (1\ 0\ 0\ 0\ \ 1\ 0\ 0\ 0\ \ 1\ 0\ 0\ 0\ \ 1\ 0\ 0\ 0)$

$f(v_{10}) = (1\ 1\ 1\ 1\ \ 0\ 0\ 0\ 0\ \ 1\ 1\ 1\ 1\ \ 0\ 0\ 0\ 0)$

$f(v_{11}) = (1\ 1\ 1\ 0\ \ 1\ 1\ 1\ 0\ \ 0\ 0\ 0\ 0\ \ 0\ 0\ 0\ 0)$

$f(v_{12}) = (0\ 0\ 0\ 1\ \ 0\ 0\ 0\ 1\ \ 0\ 0\ 0\ 1\ \ 0\ 0\ 0\ 1)$

$f(v_{13}) = (1\ 0\ 1\ 0\ \ 1\ 0\ 1\ 0\ \ 1\ 0\ 1\ 0\ \ 1\ 0\ 1\ 0)$

$f(v_{14}) = (1\ 1\ 0\ 0\ \ 1\ 1\ 0\ 0\ \ 1\ 1\ 0\ 0\ \ 1\ 1\ 0\ 0)$.

CSPF's for gates v_5, v_6, v_7, and v_8 are as follows:

$G_c(v_5) = (1\ 1\ 1\ 0\ \ 1\ 1\ 1\ 0\ \ 1\ 1\ 1\ 1\ \ 1\ 0\ 0\ 1)$,

$G_c(v_6) = (0\ 0\ 0\ *\ \ 0\ 0\ 0\ *\ \ 0\ 0\ 0\ 0\ \ 0\ *\ 1\ 0)$,

$G_c(v_7) = (0\ 0\ 0\ *\ \ 0\ 0\ 0\ *\ \ 0\ 0\ 0\ 0\ \ 0\ 1\ *\ 0)$,

$G_c(v_8) = (0\ 0\ 0\ 1\ \ 0\ 0\ 0\ 1\ \ 0\ 0\ 0\ 0\ \ 0\ *\ *\ 0)$.

Sets of connectable functions with respect to these CSPF's are:

$K(v_5) = (0\ 0\ 0\ *\ \ 0\ 0\ 0\ *\ \ 0\ 0\ 0\ 0\ \ 0\ *\ *\ 0)$,

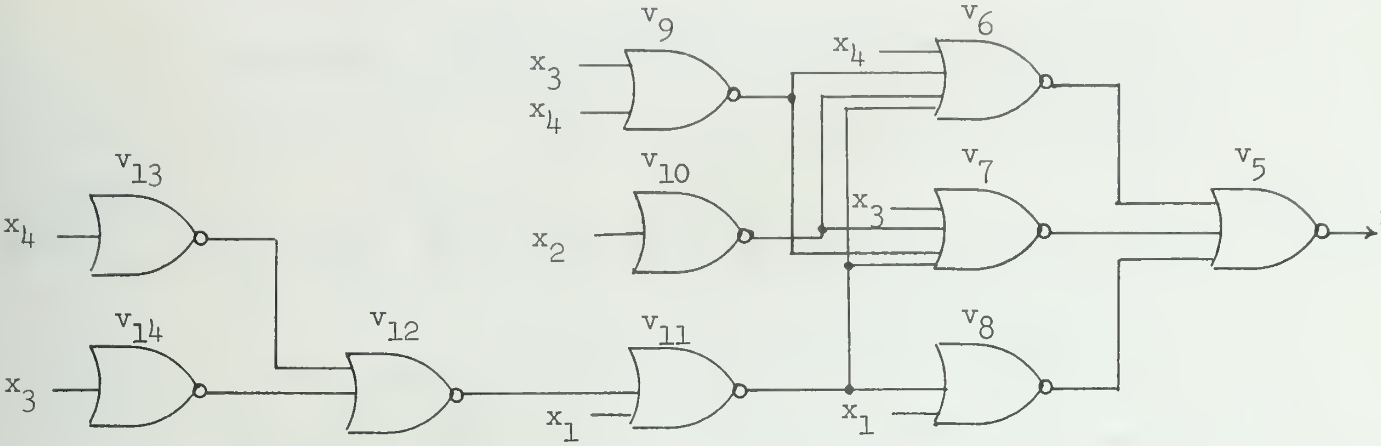
$K(v_6) = (*\ *\ *\ *\ \ *\ *\ *\ *\ \ *\ *\ *\ *\ \ *\ *\ 0\ *)$,

$K(v_7) = (*\ *\ *\ *\ \ *\ *\ *\ *\ \ *\ *\ *\ *\ \ *\ 0\ *\ *)$,

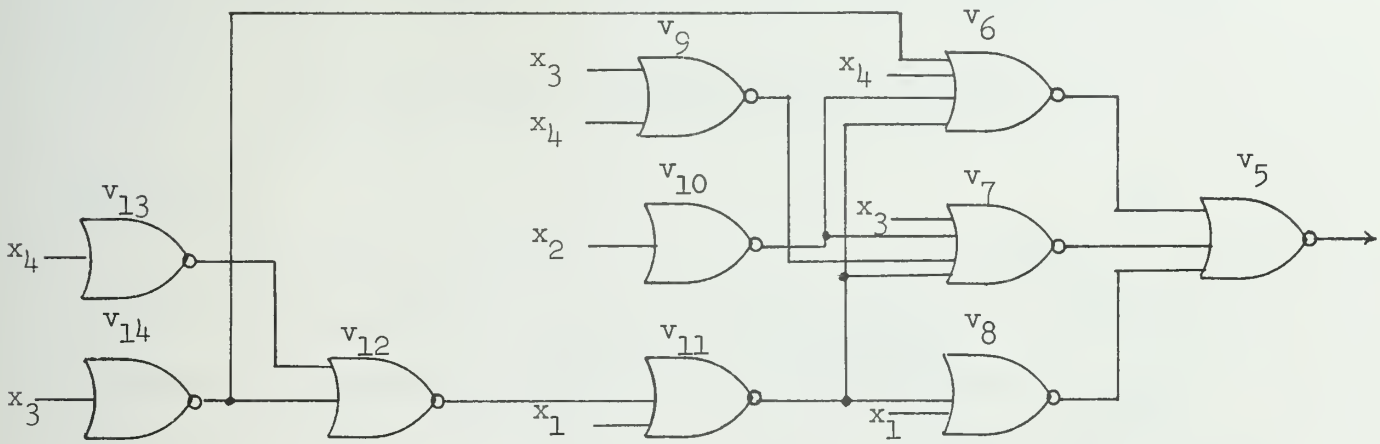
$K(v_8) = (*\ *\ *\ 0\ \ *\ *\ *\ 0\ \ *\ *\ *\ *\ \ *\ *\ *\ *)$.

These sets are used in the next few steps to transform the network.

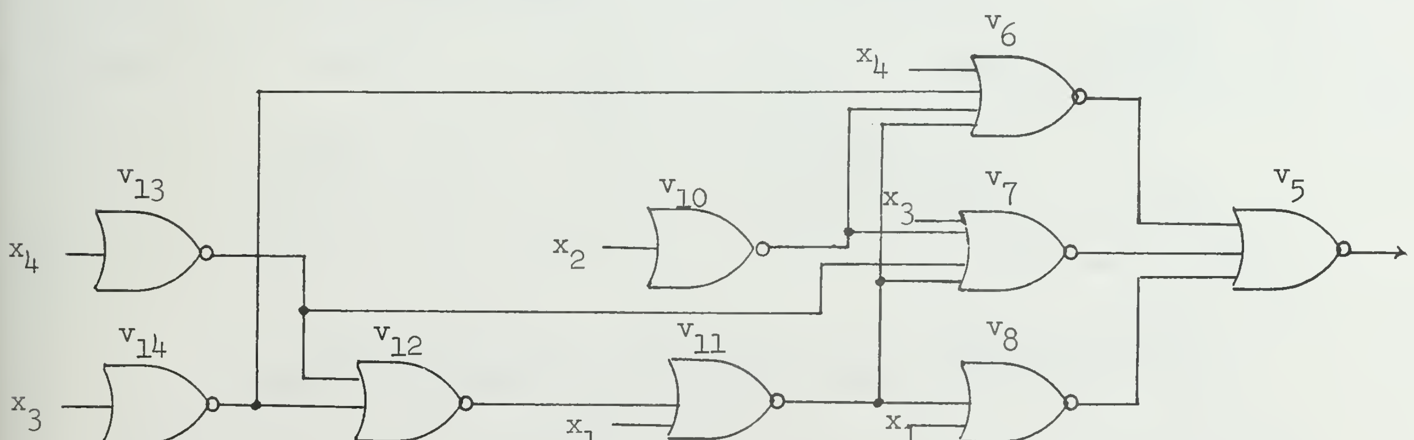
There are no gates or input terminals connectable to v_5 other than the three gates, v_6, v_7, and v_8, already connected.



(a) Given network.

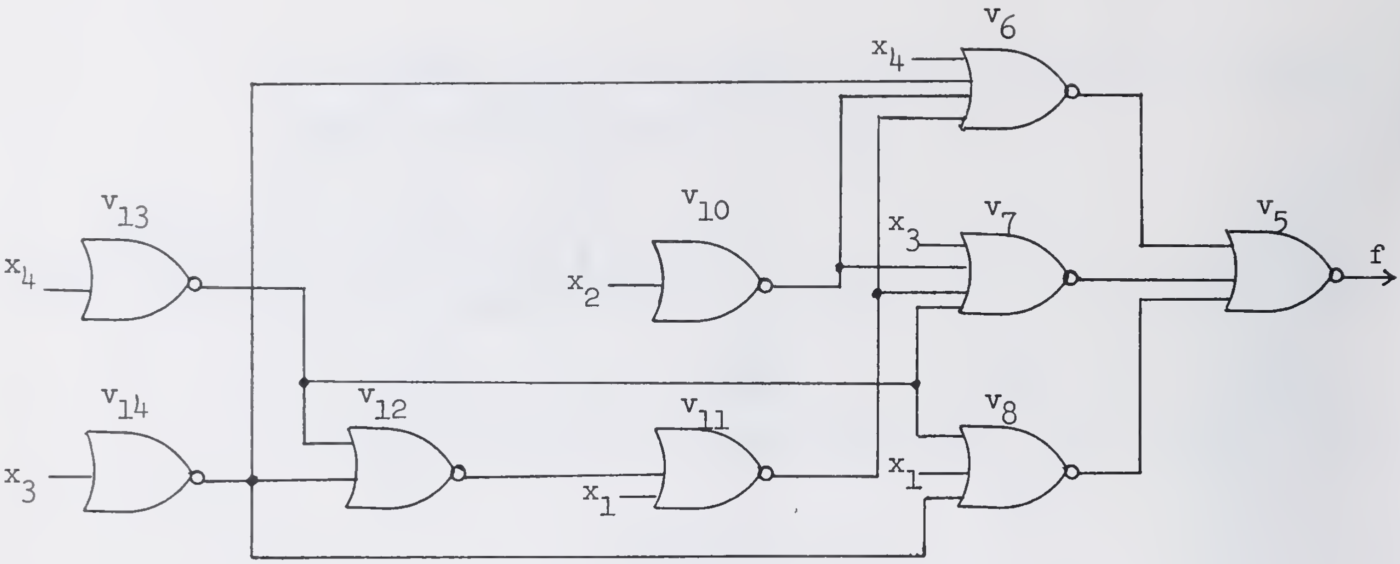


(b) Network after replacement of input set for v_6.

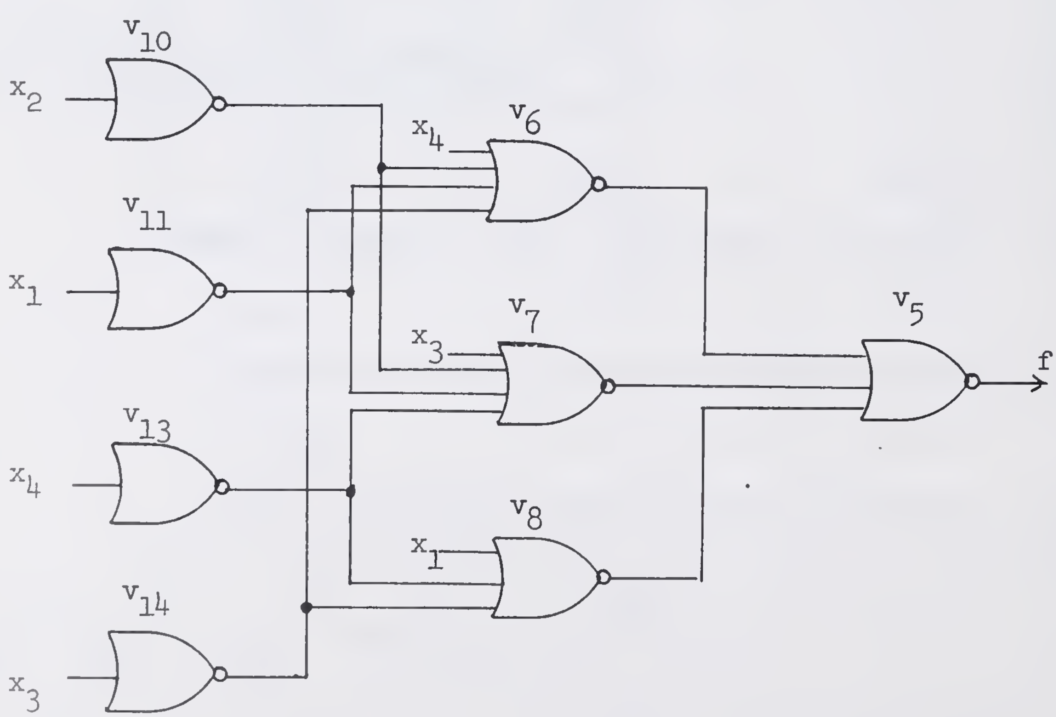


(c) Network after replacement of input set for v_7.

Fig. 3.1 Networks for Example 3.1.



(d) Network after replacement of input set for v_8.



(e) Final network — after replacement of input set for v_{11}.

Fig. 3.1 (cont.) Networks for Example 3.1.

Considering v_6, though, v_4, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, and v_{14} are all connectable to it. Furthermore, v_{10} and v_{11} are essential inputs since:

$$f^{(11)}(v_{10}) = 1,\ f^{(11)}(v_i) = 0 \text{ for } i = 4,7,8,9,11,12,14;$$

$$f^{(7)}(v_{11}) = 1,\ f^{(7)}(v_i) = 0 \text{ for } i = 4,7,8,9,10,12,14.$$

There are no other essential inputs to v_6. Since:

$$f(v_{10}) \vee f(v_{11}) = (1\ 1\ 1\ 1\quad 1\ 1\ 1\ 0\quad 1\ 1\ 1\ 1\quad 0\ 0\ 0\ 0) \text{ and}$$

$$G_c(v_6) = (0\ 0\ 0\ *\quad 0\ 0\ 0\ *\quad 0\ 0\ 0\ 0\quad 0\ *\ 1\ 0),$$

in order to realize a function in $G_c(v_6)$ an additional function is needed from the set

$$(*\ *\ *\ *\quad *\ *\ *\ *\quad *\ *\ *\ *\quad 1\ *\ 0\ 1).$$

The function

$$f(v_4) \vee f(v_{14}) = (1\ 1\ 0\ 1\quad 1\ 1\ 0\ 1\quad 1\ 1\ 0\ 1\quad 1\ 1\ 0\ 1)$$

is in the set, and the selection of v_4 and v_{14} as inputs to v_6 (in addition to the essential inputs, v_{10} and v_{11}) is consistent with the preference ordering r.

The new input set for v_6 is $\{v_4, v_{10}, v_{11}, v_{14}\}$, and the result is shown in Fig. 3.1(b). CSPF's chosen for the inputs of v_6 are as follows (covers for 0-components of $G_c(v_6)$ are selected in accordance with r):

$$G_c(c_{4,6}) = (*\ 1\ *\ *\quad *\ 1\ *\ *\quad *\ 1\ *\ 1\quad *\ *\ 0\ 1),$$

$$G_c(c_{14,6}) = (1\ *\ *\ *\quad 1\ *\ *\ *\quad 1\ *\ *\ *\quad 1\ *\ 0\ *),$$

$$G_c(c_{11,6}) = (*\ *\ 1\ *\quad *\ *\ 1\ *\quad *\ *\ *\ *\quad *\ *\ 0\ *),$$

$$G_c(c_{10,6}) = (*\ *\ *\ *\quad *\ *\ *\ *\quad *\ *\ 1\ *\quad *\ *\ 0\ *).$$

Next, v_3, v_6, v_8, v_9, v_{10}, v_{11}, v_{12}, and v_{13} are found to be connectable to v_7. By a calculation similar to that for v_6, $\{v_3, v_{10}, v_{11}, v_{13}\}$ is chosen as the input set for v_7. This change leaves v_9 with no output connections, and it can be removed, resulting in Fig. 3.1(c). CSPF's for the

input connections are chosen as follows:

$$G_c(c_{3,7}) = (* * 1 * \;\; * * 1 * \;\; * * 1\ 1 \;\; * 0 * 1),$$

$$G_c(c_{13,7}) = (1 * * * \;\; 1 * * * \;\; 1 * * * \;\; 1\ 0 * *),$$

$$G_c(c_{11,7}) = (* 1 * * \;\; * 1 * * \;\; * * * * \;\; * 0 * *),$$

$$G_c(c_{10,7}) = (* * * * \;\; * * * * \;\; * 1 * * \;\; * 0 * *).$$

Connectable input terminals and gates to v_8 are: v_1, v_6, v_7, v_9, v_{11}, v_{13}, and v_{14}. v_1, v_{13}, and v_{14} are selected as inputs to v_8, and the result is shown in Fig. 3.1(d).

Consideration of the input set for v_{10} leaves it unchanged.

The CSPF is determined for v_{11} as follows:

$$G_c(v_{11}) = G_c(c_{11,6}) \cap G_c(c_{11,7})$$
$$= (* 1\ 1 * \;\; * 1\ 1 * \;\; * * * * \;\; * 0\ 0 *).$$

Thus,

$$K(v_{11}) = (* 0\ 0 * \;\; * 0\ 0 * \;\; * * * * \;\; * * * *),$$

and only v_1, v_8, and v_{12} are connectable to v_{11}.

In order for v_{11} to realize a function in $G_c(v_{11})$, an input function is required from the set

$$(* 0\ 0 * \;\; * 0\ 0 * \;\; * * * * \;\; * 1\ 1 *).$$

Input terminal v_1 is both a member of this set and an essential input. Therefore, the connection $c_{12,11}$ is redundant.

Gate v_{12}, now without output connections, may be removed from the network. The final network is shown in Fig. 3.1(e). Two gates have been removed during this transduction procedure.

In order to facilitate the selection of input sets for gates, the concept of effective connectability is introduced.

<u>Definition 3.1</u> Input terminal or gate v_i is said to be <u>effectively</u>

connectable to gate v_j with respect to set $G(v_j)$ if and only if it is connectable and there exists at least one d satisfying:

$$f^{(d)}(v_i) = 1 \text{ and } G^{(d)}(v_j) = 0.$$

If input terminal or gate v_i is connectable but not effectively connectable to gate v_j with respect to set $G(v_j)$, then for every d such that $G^{(d)}(v_j) = 0$, $f^{(d)}(v_i) = 0$ must hold. Therefore, even if such a v_i were to be connected to v_j, the connection would not enable the enlargement of CSPF's for other inputs of v_j. For this reason, the effectively connectable condition is used advantageously in place of the connectable condition.

Theorem 3.1 If input terminal or gate v_i and gate v_j satisfy the condition:

$$IS(v_i) \supseteq IS(v_j),$$

then v_i is not effectively connectable to v_j with respect to $G_c(v_j)$ as calculated by Procedure RSCS given in [5].

Proof Assume that v_i is effectively connectable to v_j with respect to $G_c(v_j)$. Then there exists at least one d satisfying:

$$f^{(d)}(v_i) = 1 \text{ and } G_c^{(d)}(v_j) = 0.$$

Since v_i is connected to every gate v to which v_j is connected, for each such v, $f^{(d)}(v) = 0$ (see Fig. 3.2). By the use of Procedure RSCS [5] to calculate CSPF's, $G_c^{(d)}(v_j) = *$, because $f^{(d)}(v) = 0$ and $f^{(d)}(v_j) = 0$ (implied by $G_c^{(d)}(v_j) = 0$). This, of course, contradicts the original assumption that $G_c^{(d)}(v_j) = 0$.

Q.E.D.

For example, in the network of Fig. 3.3, v_{i_j} (j=1,2,3) is not effectively connectable to v_{i_k} (k=1,2,3; k≠j).† Thus, Theorem 3.1 can be used

†This corresponds to the triangular condition in [1].

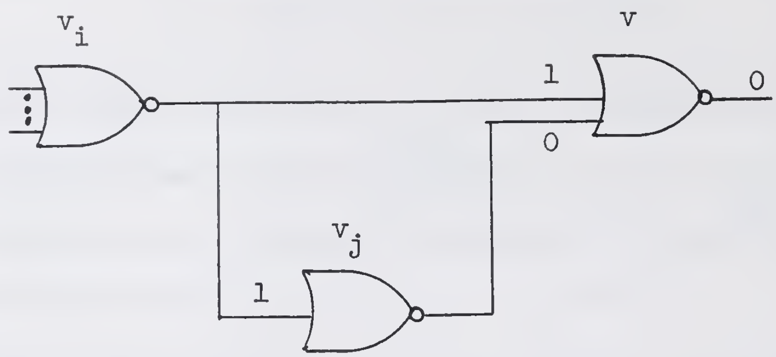


Fig. 3.2 Configuration in which v_i is not effectively connectable to v_j.

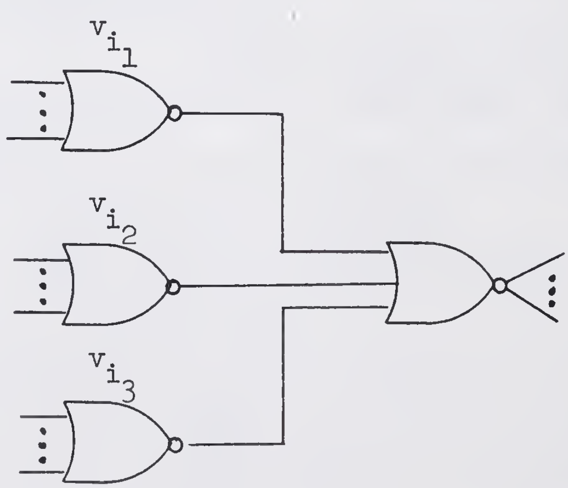


Fig. 3.3 Configuration in which no pair of gates v_{i_1}, v_{i_2}, v_{i_3} are effectively connectable.

to exclude certain candidates for effectively connectable input terminals and gates.

While determining the set of effectively connectable gates and input terminals for a gate v_j, suppose a gate v_i is encountered which is effectively connectable to v_j and whose CSPF has already been calculated. In such a case, the addition of a connection from v_i to v_j might necessitate a recalculation of the CSPF for v_i since a new output connection for v_j will have been created. In addition, if there exist gates in $P(v_i)$ whose CSPF's have already been calculated, these may also require recalculation. This process would involve excess calculation time; hence, connections are added only when no recalculation of CSPF's is necessary. A condition for this is given in the following lemma.

<u>Lemma 3.1</u> Let $G_c(v_i)$ and $G_c(v_j)$ be CSPF's for gates v_i and v_j respectively. If the following conditions are satisfied, v_i is effectively connectable to v_j and the CSPF for v_i does not change by adding a connection from v_i to v_j:

(1) For every d such that $G_c^{(d)}(v_j) = 1$, $G_c^{(d)}(v_i) = 0$.

(2) Gate v_i is connectable to v_j.

(3) There exists at least one d such that $G_c^{(d)}(v_j) = 0$ and $G_c^{(d)}(v_i) = 1$.

<u>Proof</u>: For all d such that $G_c^{(d)}(v_j) = 0$ or *, the value of $G_c^{(d)}(v_i)$ is unchanged by the addition of a connection from v_i to v_j. If there exists a d such that $G_c^{(d)}(v_j) = G_c^{(d)}(v_i) = 0$, v_i is not connectable to v_j. If there exists a d such that $G_c^{(d)}(v_j) = 1$, $f^{(d)}(v_i) = 0$ and $G_c^{(d)}(v_i) = *$, then the d-th coordinate of the vector representing the CSPF for v_i must be 0, and thus, the CSPF would change in making the new connection.

Condition (3) is a sufficient condition for the connectability of

of v_i to v_j to be effective connectability.

Q.E.D.

Ordering r affects the results of the network transduction procedure. When networks having various characteristics are desired (e.g., low cost, small maximum fanout, small number of levels, etc.), the following additional conditions (the condition that for every input terminal v_i and gate v_j, $r(v_i) > r(v_j)$, is retained in each case) may be used to determine an ordering r.

Suppose a reduction of the number of gates is of primary importance. Since it is generally more likely that a gate with smaller fan-out can be removed then a gate with larger fan-out, the following condition is imposed. For gates v_i and v_j such that $|IS(v_i)| > |IS(v_j)|$,

$$r(v_i) > r(v_j),$$

where $|IS(v_i)|$ denotes the number of elements in $IS(v_i)$. Especially, the use of a gate with only a single output connection should be avoided.

If it is desired to design a network using fan-out restricted gates, gates with smaller fan-out should be assigned larger values of r.

If a network is desired with a small number of levels, a gate whose predecessors form a subnetwork of fewer levels should have a larger value of r.

Another possibility for the purpose of reducing the number of gates is to prefer the use of gates with fewer predecessors. In other words, if v_i and v_j satisfy $|P(v_i)| > |P(v_j)|$,

$$r(v_j) > r(v_i).$$

Thus, functions which are produced by smaller subnetworks are preferred. As a simplification, the gate level of a gate can be substituted for the number

of its predecessors. A gate in a lower level (i.e., closer to the output gate(s)) generally has more predecessors than a gate in a higher level.

The following definition introduces an operation needed in the network transduction procedure to be given.

Definition 3.1 The operation Δ is defined as follows:

$$1\Delta 0 = 1\Delta 1 = 1\Delta * = 0\Delta 1 = *\Delta 1 = 1,$$

$$0\Delta 0 = 0\Delta * = *\Delta 0 = *\Delta * = 0.$$

The operation Δ which is both commutative and associative is summarized in Table 3.1.

Δ	0	1	*
0	0	1	0
1	1	1	1
*	0	1	0

Table 3.1 The operation Δ.

The network transduction procedure based on connectable and disconnectable conditions is formally given as follows:

Procedure 3.1 [Procedure NTCD] (a Network Transduction procedure based on Connectable and Disconnectable conditions)

(1) Calculate $f(v_i)$ for each gate v_i in the network.

(2) For each connection from an output gate v_i, $p < i \leq p + m$, to its corresponding output terminal v_j, $j = i + R$, $G_c(c_{ij})$ is given by:

$$G_c(c_{ij}) = z_{i-p},$$

where z_ℓ is the ℓ-th output function.

(3) Select a gate v_k for which no CSPF has yet been calculated, but for which CSPF's of all output connections have been calculated. There will always

exist at least one such v_k at this step until all gates have been exhausted. If no such gate v_k exists, go to Step (6). Determine a CSPF for gate v_k by using the equation:

$$G_c(v_k) = \bigcap_{v_j \varepsilon IS(v_k)} G_c(c_{kj}).$$

(4) Change the input set of gate v_k selected in Step (3).

(4-1) Select effectively connectable gates and input terminals for v_k.

(4-1-1) Calculate the set $K(v_k)$ of functions connectable to v_k with respect to $G_c(v_k)$.

(4-1-2) Let $Q''(v_k)$, $Q''(v_k) = Q''_1(v_k) \cup Q''_2(v_k)$, denote a set of gates and/or input terminals which represent a set of input candidates for v_k.

(4-1-3) Let $Q''_2(v_k)$ be the set of all gates satisfying Lemma 3.1 for connectability to v_k.

(4-1-4) Initially, let $Q''_1(v_k)$ be the set of all input terminals which are effectively connectable to v_k.

(4-1-5) Add to $Q''_1(v_k)$ all gates which are effectively connectable to v_k and whose CSPF's have not yet been calculated.

(4-2) If there exists a single-output gate v_i which is originally connected to v_k (i.e., $v_i \varepsilon IP(v_k)$) and which satisfies the following condition, remove v_i from $Q''(v_k)$ and from the network: For every d such that $G_c^{(d)}(v_k) = 0$,

$$\bigvee_{\substack{v \varepsilon Q''_1(v_k) \\ v \neq v_i}} f^{(d)}(v) \quad \vee \quad \Delta_{v \varepsilon Q''_2(v_k)} G_c^{(d)}(v) = 1.$$

Apply this test to each single-output gate in $IP(v_k)$.

(4-3) Determine a set, $Q_o(v_k)$, of essential inputs for v_k.

(4-4) Select a new input set Q' for v_k by the following method. If $Q' \neq IP(v_k)$, then the ordering r must be recalculated.†

†For certain orderings r, skipping this recalculation would not be too harmful.

(4-4-1) Initially, let $Q' = Q_o(v_k)$.

(4-4-2) Add every v_i, $v_i \varepsilon Q''_1(v_k)$, to Q' which satisfies the following conditions: $v_i \notin Q'$ and for at least one d such that $G_c^{(d)}(v_k) = 0$,

$$f^{(d)}(v_i) = 1 \text{ and } \bigvee_{\substack{v \varepsilon Q''_1(v_k) \\ v \neq v_i}} f^{(d)}(v) \quad \vee \quad \Delta_{v \varepsilon Q''_2(v_k)} G_c^{(d)}(v) = 0.$$

(4-4-3) If for every d such that $G^{(d)}(v_k) = 0$,

$$\bigvee_{v \varepsilon Q' \cap Q''_1(v_k)} f^{(d)}(v) \quad \vee \quad \Delta_{v \varepsilon Q' \cap Q''_2(v_k)} G_c^{(d)}(v) = 1$$

is satisfied, recalculate ordering r if Q' is different from the original input set for v_k and go to Step (5).

(4-4-4) Add v_i to Q' if v_i satisfies the following conditions: $v_i \varepsilon Q''(v_k)$, v_i is not currently a member of Q', and there exists at least one d such that $G_c^{(d)}(v_k) = 0$; $f^{(d)}(v_i) = 1$ if $v_i \varepsilon Q''_1(v_k)$ or $G_c^{(d)}(v_i) = 1$ if $v_i \varepsilon Q''(v_k)$; and

$$\bigvee_{v \varepsilon Q' \cap Q''_1(v_k)} f^{(d)}(v) \quad \vee \quad \Delta_{v \varepsilon Q' \cap Q''_2(v_k)} G_c^{(d)}(v) = 0.$$

The v_i selected is the one with largest value of $r(v_i)$ among those satisfying these conditions. Go to Step (4-4-3).

(5) Calculate CSPF's for connections from gates and input terminals in Q' to v_k by the following method.

(5-1) Let $H = Q''_2(v_k) \cap Q'$. Calculate a vector G as follows:

If $G_c^{(d)}(v_k) = *$, set $G^{(d)} = *$.

If $G_c^{(d)}(v_k) = 1$, set $G^{(d)} = 1$.

If $G_c^{(d)}(v_k) = 0$ and $G_c^{(d)}(v) = 1$ for at least one $v \varepsilon H$, set $G^{(d)} = *$.

Otherwise, set $G^{(d)} = 0$.

After connections from gates in H have been added to v_k, G can be regarded as representing a CSPF for v_k for the remaining inputs $(Q' - H)$.

(5-2) Calculate $G_c(c_{ik})^{\dagger}$ for $v_i \varepsilon Q'-H$ by the following equation:

$$G_c(c_{ik}) = \{G \square (\bigvee_{\substack{r(v)>r(v_i) \\ v\varepsilon Q'-H}} f(v))\} \# f(v_i)$$

Go to Step (3).

(6) Recalculate $f(v_i)$ for each gate v_ℓ remaining in the network. If there exist v_i and v_j such that

$$f(v_i) \varepsilon G_c(v_j) \text{ and } v_i \not\in S(v_j),$$

then replace output connections from v_j by output connections from v_i, removing v_j from the network. If successful, recalculate CSPF's for gates in the network and repeat this step.

This procedure does not calculate CSPF's for input terminals since they are generally not useful. However, if they are desired, a new step - similar to Step (3) for gates - can be inserted prior to Step (6) to accomplish this.

In Procedure NTCD, Q' is determined by the ordering r. Following are two alternative selection criteria:

(1) Remove as many originally connected gates and input terminals as possible.

(2) Select Q' which has the minimum number of elements.

Selection criterion (1) is used to attempt to produce a relatively large change in a network's configuration. By changing a network's configuration, other known network transduction procedures may be able to be applied which could not be effectively applied to the original network. (A similar criterion was incorporated in the computer program NETTRA-G1 described in Section 5.) Selection criterion (2) can be used in network design with fan-in restricted gates.

---

†Normally, $G_i(c_{ik})$ need not be calculated if v_i is an input terminal.

A relatively simple additional calculation can be performed in Step (5) of Procedure NTCD to produce generally larger CSPF's. Suppose a CSPF has already been determined for at least one output connection, c_{ij}, of a gate v_i. If for some d, $G_c^{(d)}(c_{ij}) = 1$, then $G_c^{(d)}(v_i) = 1$ is known even though CSPF's for other output connections of v_i may not yet have been calculated. Thus, in Step (5), if there exists any output connection, c_{ij}, of any input v_i of v_k for which a CSPF, $G_c(c_{ij})$, has already been calculated, and $G_c^{(d)}(c_{ij}) = 1$ for some d for which $G_c^{(d)}(v_k) = 0$, then the d-th coordinate of every vector representing a CSPF for some input connection to v_k can be assigned the value *.

This is accomplished by the addition of the following condition to those listed in Step (5-1).

If $G_c^{(d)}(v_k) = 0$ and $G_c^{(d)}(c_{ij}) = 1$ has been calculated for at least one output connection, c_{ij}, of v_i where $v_i \varepsilon Q'$, set $G^{(d)} = *$.

4. MODIFICATION OF THE NETWORK TRANSDUCTION PROCEDURE

In this section, a modification of Procedure NTCD is given which focuses on the removal of a specific gate of a network.

The following notation will facilitate explanation of the modified procedure. $T(v_\ell)$ represents the set of all gates which are neither successors of v_ℓ nor v_ℓ itself:

$$T(v_\ell) = V_G - S(v_\ell) - \{v_\ell\}.$$

This modification of Procedure NTCD seeks the removal of a specific gate, v_ℓ, by first attempting to enlarge, to a maximum extent, the set of permissible functions for v_ℓ. This increases the probability that one of the following will occur:

(1) $G_c(v_\ell)$ will contain some function $f(v)$ $(v \varepsilon T(v_\ell) \cup V_I)$. In such a case, v_ℓ can be replaced by v.

(2) For all d, $G_c^{(d)}(v_\ell) \neq 1$. In this case, $G_c(v_\ell)$ contains the function which is always 0, and v_ℓ can be removed from the network.

This is essentially accomplished by adding connections from input terminals and gates in $T(v_\ell)$ to gates in $S(v_\ell)$ in order to replace as many input connections from other gates in $S(v_\ell)$ as possible and to increase the size of the CSPF's for those connections which cannot be replaced. Basically, this involves a modification of Step (4) of Procedure NTCD and the selection of a special type of ordering r.

In the following transduction procedure, an ordering r satisfying the conditions below is assumed.

(1) $1 \leq r(v_i) \leq R$ for $v_i \varepsilon V_G$,

$R+1 \leq r(v_i) \leq R+n$ for $v_i \varepsilon V_I$.

(2) For every gate v_i in $T(v_\ell)$ and every gate v_k in $S(v_\ell) \cup \{v_\ell\}$,

$$r(v_i) > r(v_k).$$

Procedure 4.1 [Procedure NTCDG] (A Network Transduction procedure, based on Connectable and Disconnectable conditions, modified to concentrate on the removal of a specific Gate)

(0) Select gate v_ℓ. Calculate $S(v_\ell)$ and $T(v_\ell)$ and select an ordering r. Assume $S(v_\ell) \neq \phi$. Do not alter $S(v_\ell)$ or $T(v_\ell)$ during the remainder of the procedure.

(1) Calculate $f(v_i)$ for each gate v_i in the network.

(2) For each connection from an output gate v_i belonging to $S(v_\ell) \cup \{v_\ell\}$, $p < i \leq p+m$, to its corresponding output terminal v_j, $j = i+R$, $G_c(c_{ij})$ is given by:

$$G_c(c_{ij}) = z_{i-p},$$

where z_{i-p} is the (i-p)-th output function.

(3) Select a gate v_k in $S(v_\ell) \cup \{v_\ell\}$ for which no CSPF has been calculated, but for which CSPF's of all output connections have been calculated. There will always exist at least one such v_k at this step until all gates in $S(v_\ell) \cup \{v_\ell\}$ have been exhausted. Determine a CSPF for gate v_k by using the equation:

$$G_c(v_k) = \bigcap_{v_i \varepsilon IS(v_k)} G_c(c_{kj}).$$

If $k = \ell$, go to Step (6).

(4) Change the input set of gate v_k selected in Step (3).

(4-1) Select effectively connectable gates in $T(v_\ell)$ and effectively connectable input terminals for v_k with respect to $G_c(v_k)$. Connect all such gates and input terminals to v_k.

(4-2) If there exists a single-output gate v_i in $S(v_\ell) \cup \{v_\ell\}$ which is connected to v_k (i.e., $v_i \varepsilon IP(v_k)$) and which satisfies the condition (Theorem 2.2) for

disconnectability from v_k with respect to $G_c(v_k)$, then disconnect v_i from v_k and remove it from the network. Apply this test to each single-output gate in $IP(v_k) - T(v_\ell)$.

(4-3) For each gate in $S(v_\ell) \cup \{v_\ell\}$ which is connected to v_k and which is not a single-output gate, check whether or not it satisfies the condition (Theorem 2.2) for disconnectability from v_k with respect to $G_c(v_k)$. If so, disconnect it from v_k. Selection priority for this test is the reverse of ordering r.

(4-4). If a gate in $T(v_\ell)$ or an input terminal, v_i, which is connected to v_k satisfies the following condition, remove the connection from v_i to v_k: For every d such that $G^{(d)}(v_k) = 0$ and $f^{(d)}(v_i) = 1$,

$$\bigvee_{\substack{v \in IP(v_k) \cap [T(v_\ell) \cup V_I] \\ v \neq v_i}} f^{(d)}(v) = 1$$

Selection priority for this test is the reverse of ordering r.

(5) Let Q' denote the set of inputs for v_k after the completion of Steps (4-1) to (4-4). Calculate CSPF's for connections from gates in $[S(v_\ell) \cup \{v_\ell\}] \cap Q'$ to v_k by the following method.

(5-1) Let H be the set of input terminals and gates which are in Q' but not in $S(v_\ell) \cup \{v_\ell\}$. Calculate a vector G as follows:

If $G_c^{(d)}(v_k) = *$, set $G^{(d)} = *$.

If $G_c^{(d)}(v_k) = 1$, set $G^{(d)} = 1$.

If $G_c^{(d)}(v_k) = 0$ and $f^{(d)}(v) = 1$ for at least one $v \varepsilon H$, set $G^{(d)} = *$.

Otherwise, set $G^{(d)} = 0$.

For inputs from gates in $S(v_\ell) \cup \{v_\ell\}$, G can be regarded as a CSPF for v_k.

(5-2) Calculate $G_c(c_{ik})$ for $v_i \varepsilon S(v_\ell) \cup \{v_\ell\}$ by the following equation:

$$G_c(c_{ik}) = \{G \square (\bigvee_{\substack{r(v) > r(v_i) \\ v \varepsilon Q' - H}} f(v))\} \# f(v_i).$$

Go to Step (3).

(6) If there exists v_i in $T(v_\ell) \cup V_I$ satisfying

$$f(v_i) \in G_c(v_\ell)$$

v_ℓ can be replaced by v_i. If v_ℓ has no output connections, it can be removed from the network.

At several points in Procedure NTCDG, variations exist to the course of the procedure as formulated above. Incorporation of such variations into the procedure can alter the emphasis (e.g., the above formulation of the procedure places a relatively high value on the removal of gate v_ℓ and the ease of computation, a relatively lower value on the attainment of a final network with few redundant connections, and no value at all on the calculation of CSPF's for gates in $T(v_\ell)$) of the transduction. Three examples of possible variations in the procedure are given below. The corresponding changes in emphasis of the transduction are obvious.

A. Step (4-2) can be followed by a similar step for attempting the removal of single-output gates in $T(v_\ell)$ connected to v_k. If successful, this would result in the immediate removal of one gate from the network, although the possibility of removing v_ℓ may be adversely affected as a consequence.

B. Somewhere in Step (4), a test can be made to determine if any single-output gates in $T(v_\ell)$ connected to v_k can be removed by the addition of connections to v_k from gates in $S(v_\ell) \cup \{v_\ell\}$. This would have consequences similar to those for (A).

C. Sets $S(v_\ell)$ and $T(v_\ell)$ can be kept updated during the transduction. Once v_ℓ has been selected in Step (0), the above Procedure NTCDG assumes, in the interest of computational simplicity, that $S(v_\ell)$ and $T(v_\ell)$ will remain unchanged. In actuality, however, gates which are originally successors of v_ℓ at the beginning of the transduction may no longer be such upon its termination. It is convenient not to alter sets $S(v_\ell)$ and $T(v_\ell)$ during the transduction (since ordering r and the functions of certain gates must also be updated)

but the performance of this update will add to the potential power of the transduction with respect to the removal of gate v_ℓ (since the number of functions available from the members of $T(v_\ell)$ is increased). If desired, the update should be done at the beginning of Step (5) whenever the condition

$$[S(v_\ell) \cup \{v_\ell\}] \cap Q' = \phi$$

is detected following Step (4).

Another possibility is to carry out the calculation of CSPF's for gates in $T(v_j)$ also. The actual computer program implementation of the procedure permits this option.

The following example demonstrates the application of Procedure NTCDG.

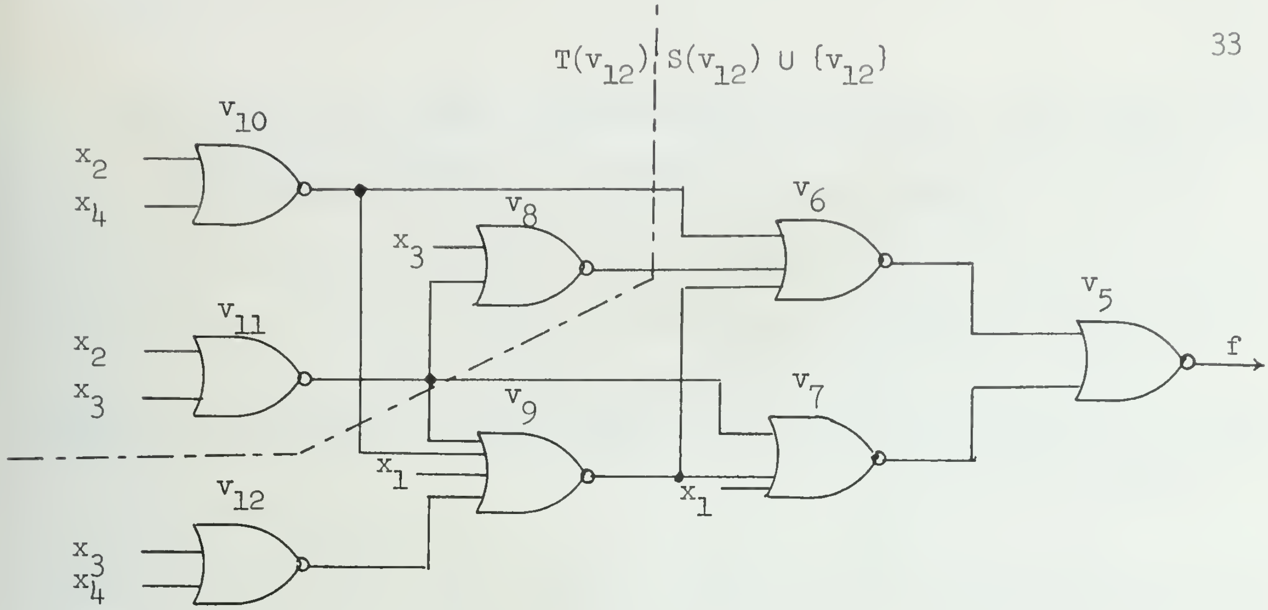
Example 4.1 The network in Fig. 4.1(a) realizes the following 4-variable function f:

$$f = \bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_4 \vee x_1x_2\bar{x}_3 \vee \bar{x}_1x_2x_4 \vee \bar{x}_1x_3x_4 \vee \bar{x}_1x_2x_3.$$

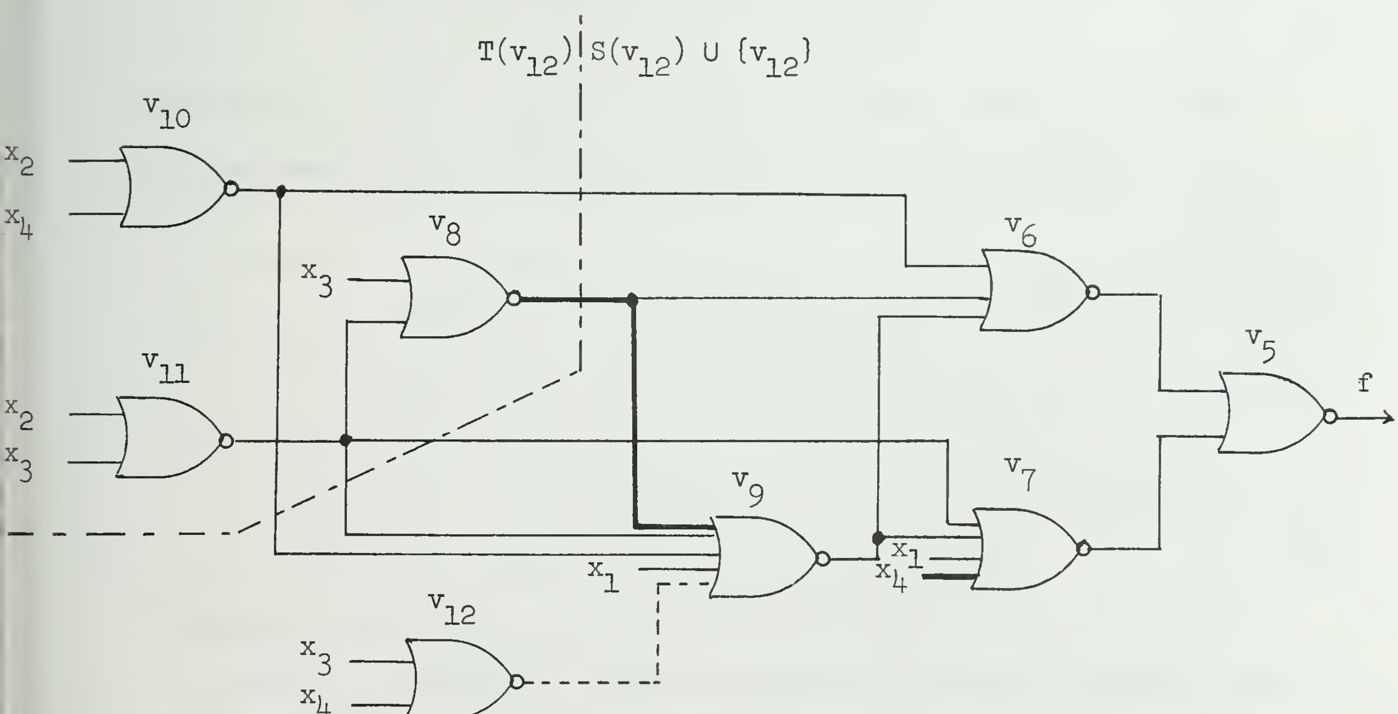
It consists of 8 gates and 20 connections.

The functions realized at gates and input terminals are as follows:

$$
\begin{aligned}
f(v_1) &= x_1 = (0\ 0\ 0\ 0\quad 0\ 0\ 0\ 0\quad 1\ 1\ 1\ 1\quad 1\ 1\ 1\ 1)\\
f(v_2) &= x_2 = (0\ 0\ 0\ 0\quad 1\ 1\ 1\ 1\quad 0\ 0\ 0\ 0\quad 1\ 1\ 1\ 1)\\
f(v_3) &= x_3 = (0\ 0\ 1\ 1\quad 0\ 0\ 1\ 1\quad 0\ 0\ 1\ 1\quad 0\ 0\ 1\ 1)\\
f(v_4) &= x_4 = (0\ 1\ 0\ 1\quad 0\ 1\ 0\ 1\quad 0\ 1\ 0\ 1\quad 0\ 1\ 0\ 1)\\
f(v_5) &= (1\ 0\ 0\ 1\quad 0\ 1\ 1\ 1\quad 1\ 0\ 1\ 0\quad 1\ 1\ 0\ 0)\\
f(v_6) &= (0\ 1\ 0\ 0\quad 0\ 0\ 0\ 0\quad 0\ 1\ 0\ 1\quad 0\ 0\ 1\ 1)\\
f(v_7) &= (0\ 0\ 1\ 0\quad 1\ 0\ 0\ 0\quad 0\ 0\ 0\ 0\quad 0\ 0\ 0\ 0)\\
f(v_8) &= (0\ 0\ 0\ 0\quad 1\ 1\ 0\ 0\quad 0\ 0\ 0\ 0\quad 1\ 1\ 0\ 0)\\
f(v_9) &= (0\ 0\ 0\ 1\quad 0\ 1\ 1\ 1\quad 0\ 0\ 0\ 0\quad 0\ 0\ 0\ 0)\\
f(v_{10}) &= (1\ 0\ 1\ 0\quad 0\ 0\ 0\ 0\quad 1\ 0\ 1\ 0\quad 0\ 0\ 0\ 0)\\
f(v_{11}) &= (1\ 1\ 0\ 0\quad 0\ 0\ 0\ 0\quad 1\ 1\ 0\ 0\quad 0\ 0\ 0\ 0)\\
f(v_{12}) &= (1\ 0\ 0\ 0\quad 1\ 0\ 0\ 0\quad 1\ 0\ 0\ 0\quad 1\ 0\ 0\ 0).
\end{aligned}
$$



(a) Given network.



(b) Final network – after use of Procedure NTCDG (gate v_{12} is isolated).

NEW CONNECTION
----- REMOVED CONNECTION

Fig. 4.1 Networks for Example 4.1.

(0) Select gate v_{12} as v_ℓ of Procedure NTCDG. $S(v_{12}) = \{v_5, v_6, v_7, v_9\}$. $T(v_{12}) = \{v_8, v_{10}, v_{11}\}$. Let ordering r be selected as follows:

$$r(v_i) = 8 + i \quad \text{for} \quad i = 1, 2, 3, 4;$$

$$r(v_5) = 1,\ r(v_6) = 2,\ r(v_7) = 3,\ r(v_9) = 4,$$

$$r(v_{12}) = 5,\ r(v_8) = 6,\ r(v_{10}) = 7,\ r(v_{11}) = 8.$$

(2) The only gate (in $S(v_{12}) \cup \{v_{12}\}$) whose output is connected to an output terminal (v_{13}) is v_5.

$$G_c(c_{5,13}) = (1\ 0\ 0\ 1 \quad 0\ 1\ 1\ 1 \quad 1\ 0\ 1\ 0 \quad 1\ 1\ 0\ 0) = f.$$

(3) v_5 is the only gate with CSPF's calculated for all of its output connections.

$$G_c(v_5) = G_c(c_{5,13}) = (1\ 0\ 0\ 1 \quad 0\ 1\ 1\ 1 \quad 1\ 0\ 1\ 0 \quad 1\ 1\ 0\ 0).$$

(4) $$K(v_5) = (0\ *\ *\ 0 \quad *\ 0\ 0\ 0 \quad 0\ *\ 0\ * \quad 0\ 0\ *\ *).$$

There exist no input terminals or gates in $T(v_{12})$ which are connectable to v_5. Neither v_6 nor v_7 satisfies the condition for disconnectability from v_5.

(5) Calculate CSPF's for connections from v_6 and v_7 to v_5.

$$G_c(c_{6,5}) = (0\ 1\ *\ 0 \quad *\ 0\ 0\ 0 \quad 0\ 1\ 0\ 1 \quad 0\ 0\ 1\ 1)$$

$$G_c(c_{7,5}) = (0\ *\ 1\ 0 \quad 1\ 0\ 0\ 0 \quad 0\ *\ 0\ * \quad 0\ 0\ *\ *)$$

(3)' $$G_c(v_6) = G_c(c_{6,5}) = (0\ 1\ *\ 0 \quad *\ 0\ 0\ 0 \quad 0\ 1\ 0\ 1 \quad 0\ 0\ 1\ 1)$$

(4)' $$K(v_6) = (*\ 0\ *\ * \quad *\ *\ *\ * \quad *\ 0\ *\ 0 \quad *\ *\ 0\ 0)$$

There exist no input terminals or gates in $T(v_{12})$ which are connectable to v_6 except v_8 and v_{10} (already connected). None of v_8, v_9, or v_{10} satifies any of the conditions for disconnectability from v_6 (Steps (4-2), (4-3), and (4-4)).

(5)' Calculate CSPF for the connection from v_9 to v_6.

$$G_c(c_{9,6}) = (*\ 0\ *\ 1 \quad *\ *\ 1\ 1 \quad *\ 0\ *\ 0 \quad *\ *\ 0\ 0)$$

(3)" $$G_c(v_7) = G_c(c_{7,5}) = (0\ *\ 1\ 0 \quad 1\ 0\ 0\ 0 \quad 0\ *\ 0\ * \quad 0\ 0\ *\ *)$$

(4)" $$K(v_7) = (*\ *\ 0\ * \quad 0\ *\ *\ * \quad *\ *\ *\ * \quad *\ *\ *\ *)$$

Since $f(v_4) = (0\ 1\ 0\ 1\ \ 0\ 1\ 0\ 1\ \ 0\ 1\ 0\ 1\ \ 0\ 1\ 0\ 1)$, v_4 is effectively connectable to v_7.No other connectable input terminals or gates in $T(v_{12})$ exist which are not already connected to v_7. Connect v_4 to v_7. Gate v_9 in $S(v_{12}) \cup \{v_{12}\}$ does not satisfy the condition in Step (4-3) for disconnectability from v_7. Input terminals v_1 and v_4 and gate $v_{11} \varepsilon T(v_{12})$ also do not satisfy the condition in Step (4-4) for disconnectability from v_7.

(5)" $Q' = \{v_i, v_4, v_9, v_{11}\}$. Calculate CSPF for the connection from v_9 to v_7.

$$G_c(c_{9,7}) = (*\ *\ 0\ *\ \ 0\ *\ 1\ *\ \ *\ *\ *\ *\ \ *\ *\ *\ *)$$

(3)"' $G_c(c_{9,6}) \cap G_c(c_{9,7}) =$

$$G_c(v_9) = (*\ 0\ 0\ 1\ \ 0\ *\ 1\ 1\ \ *\ 0\ *\ 0\ \ *\ *\ 0\ 0)$$

(4)"'

$$K(v_9) = (*\ *\ *\ 0\ \ *\ *\ 0\ 0\ \ *\ *\ *\ *\ \ *\ *\ *\ *)$$

Since $f(v_8) = (0\ 0\ 0\ 0\ \ 1\ 1\ 0\ 0\ \ 0\ 0\ 0\ 0\ \ 1\ 1\ 0\ 0)$,

v_8 is effectively connectable to v_9. No other connectable input terminals or gates in $T(v_{12})$ exist which are not already connected to v_9. Connect v_8 to v_9. In Step (4-2), connection $c_{12,9}$ is found to satisfy Theorem 2.2 for disconnectability with respect to $G_c(v_9)$: For all d such that $f^{(d)}(v_{12}) = 1$, either $G_c(v_9) = *$ or $f^{(d)}(v_1) \vee f^{(d)}(v_8) \vee f^{(d)}(v_{10}) \vee f^{(d)}(v_{11}) = 1$. Remove connection $c_{12,9}$. None of v_1, v_8, v_{10}, or v_{11} are found to be disconnectable from v_9 in Step (4-4).

As a result of Step (4)"', the network shown in Fig. 4.1(b) is obtained which consists of 7 gates and 19 connections. Furthermore, the desired gate v_{12} has been removed from the network.

5. COMPUTER PROGRAMS AND EXPERIMENTS

This chapter will discuss procedures NTCD and NTCDG as they have been actually programmed. The programs NETTRA-G1 and NETTRA-G2 realize the procedures NTCD and NTCDG, respectively.

As previously stated, the procedures as programmed are slightly different from the procedures as they were specified earlier. This is due to various reasons: programming convenience, considerations of computation time vs. effectiveness, choosing specific methods where only general methods had been specified, etc.

Each program is complete in itself: reading in the given network, applying the desired transduction procedure, and printing out the resultant network. The details necessary to use NETTRA-G1 and -G2 can be found in [2].

The procedures realized by these two programs will now be described in detail. For each program, examples will be given of its capabilities in reducing the cost of a network and/or changing the configuration of a network.

5.1 A Program Realizing Procedure NTCD

In the program NETTRA-G1, it is the subroutine named PROCII which essentially realizes Procedure NTCD. The general flowchart of this subroutine is shown in Fig. 5.1.1.

For simplicity of the explanation, it will continue to be assumed that only n uncomplemented variables are available to the given network. This means p=n, and let v_i correspond to x_i (for i=1,...,n) (i.e., "input terminal" is synonymous with "external variable").

In block 1 of the flowchart, the output functions of all of the gates, $f(v_i)$ for $i=n+1, n+2,..., n+R$, are calculated in a straightforward manner. For all non-output gates, $v_i, i=n+m+1,n+m+2,...,n+R$ (i.e., gates with

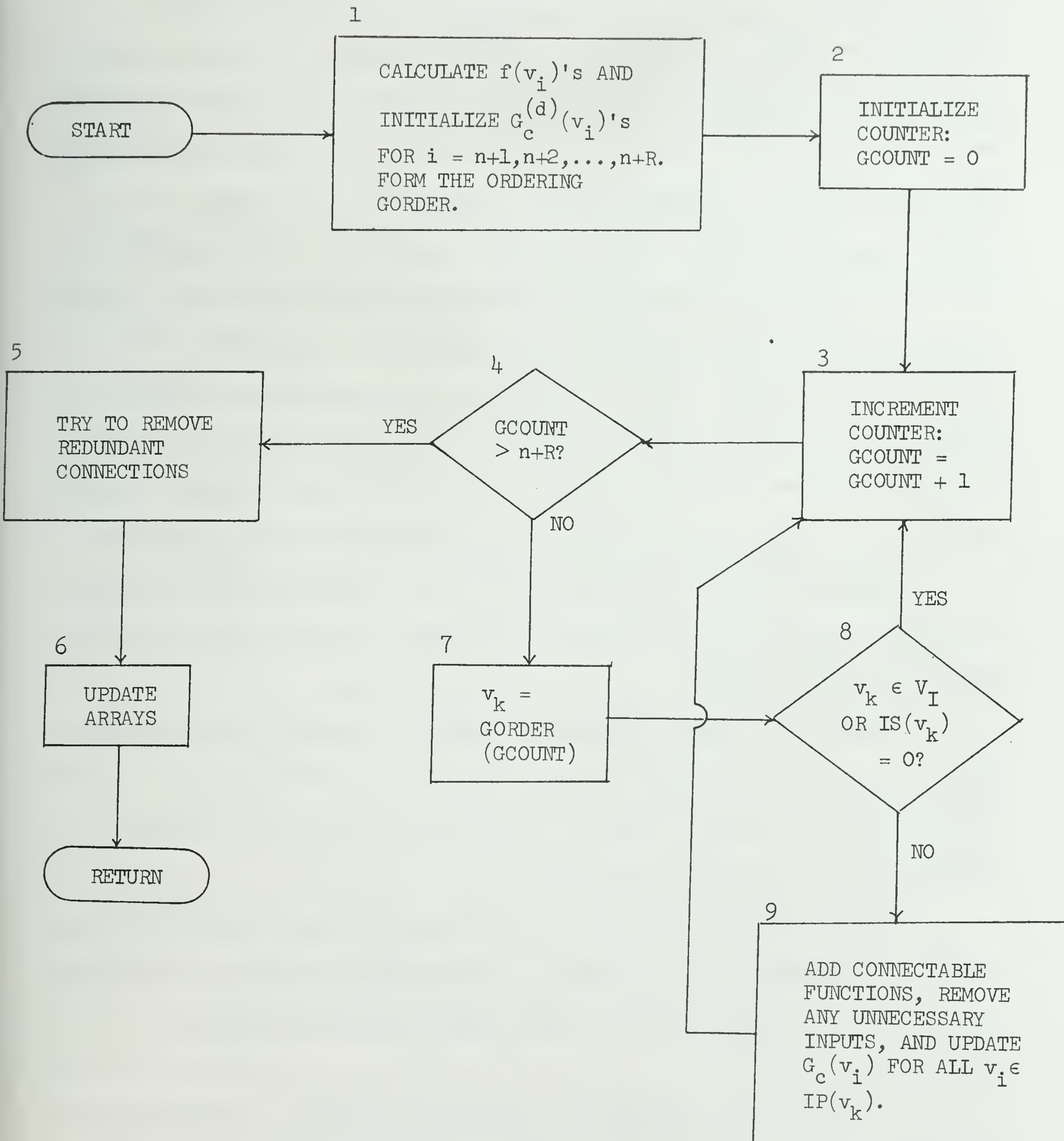


Fig. 5.1.1 General flowchart of program realizing Procedure NTCD.

no connections to any output terminal), set $G_c^{(d)}(v_i) = *$ for $d=1,2,\ldots,2^n$. For all output gates, $v_{n+j}, j=1,\ldots,m$ (i.e., gates with at least one connection to the output terminals), set $G_c^{(d)}(v_{n+j}) = z_j^{(d)}$ for $d=1,2,\ldots,2^n$. Block 1 also determines an ordering (which has a somewhat different purpose than the ordering r mentioned earlier) of gates and input terminals and stores it in an array GØRDER such that for v_i=GØRDER(i') and v_j = GØRDER(j'), i'<j' means that v_i precedes v_j in the ordering. The ordering represented by the array GØRDER satisfies the following criterion: for every gate or input terminal, v_i, and one of its successors, v_j, v_j precedes v_i in the ordering. By applying Steps (3), (4), and (5) of Procedure NTCD (see Section 3) to gates GØRDER(1),GØRDER(2),..., GØRDER(n+R) successively (ignoring GORDER(i)'sϵV_I), one is assured of always knowing the CSPF completely for a gate v_k before these steps consider it in the procedure (a necessary condition for a valid implementation of the procedure).

Block 2 simply initializes a counter, the variable GCØUNT, used in the program loop consisting of blocks 3,4,7,8,and 9. GCØUNT points to positions in the array GØRDER.

Block 3 increments the counter, GCØUNT. The program loop formed by blocks 3,4,7,8, and 9 is executed once for each value of GCØUNT: 1,2,3,... (this corresponds to the repeated execution of Steps (3), (4), and (5) of Procedure NTCD for each gate of the network).

When GCØUNT is incremented beyond the value n+R, it is detected in block 4. This signals the termination of the procedure, every gate in the network having been scanned; and the program enters block 5. Otherwise, the program proceeds to block 7.

In block 5 the procedure has essentially finished. A subroutine is called which quickly searches for and removes certain redundant connections which may still remain in the network (for example, certain new connections that might have been added unnecessarily in block 9). The removal of such

connections will not cause a change in the output functions of the network.

At the end of every transformation there are certain "house-keeping" chores which must be performed : updating or restoring values in arrays and variables. This task is done in block 6. This is followed by a return to the calling subroutine.

In block 7, v_k is the gate or input terminal whose label is contained in the (GCØUNT)$^{\underline{th}}$ position of the ordering stored in the array GØRDER. This v_k is the gate or input terminal about to be scanned by the program.

If v_k happens to be either an input terminal ($v_k \varepsilon V_I$) or an isolated gate (a gate with no successors : $IS(v_k)=\phi$), no action needs to be taken. In such a case block 8 returns control to block 3. Otherwise, the program continues to block 9.

Block 9 performs many functions and is actually quite complex. So block 9 is detailed in Fig. 5.1.2 as consisting of sub-blocks 10 through 23. Block 9 corresponds to Steps (4) and (5) of Procedure NTCD, although superficially it may first appear completely different. The variable v_k here corresponds to the v_k chosen in Step (3) of Procedure NTCD.

In block 9, a set Q' is not developed explicitly as in Step (4-4) of the procedure. New inputs are added one at a time to gate v_k by the program, and old inputs are removed one by one (when it is possible to remove them). However, the end effect, of course, is the same as constructing a set Q'(permitting $Q' \cap IP(v_k)$ to be non-null) and replacing the set $IP(v_k)$ by that Q'.

Another difference between block 9 of the program and Step (5) of the procedure is that block 9 does not calculate $G_c(c_{ij})$'s. In place of the calculation of $G_c(c_{ik})$'s, for all i such that $v_i \varepsilon IP(v_k)$, in Step (5-2) of the procedure, block 9 assigns values (0,1,or"no change") directly to components of the CSPF's of all v_i such that $v_i \varepsilon IP(v_k)$. It is thus possible to bypass the calculations of the CSPF vectors for connections since the CSPF vector of a gate

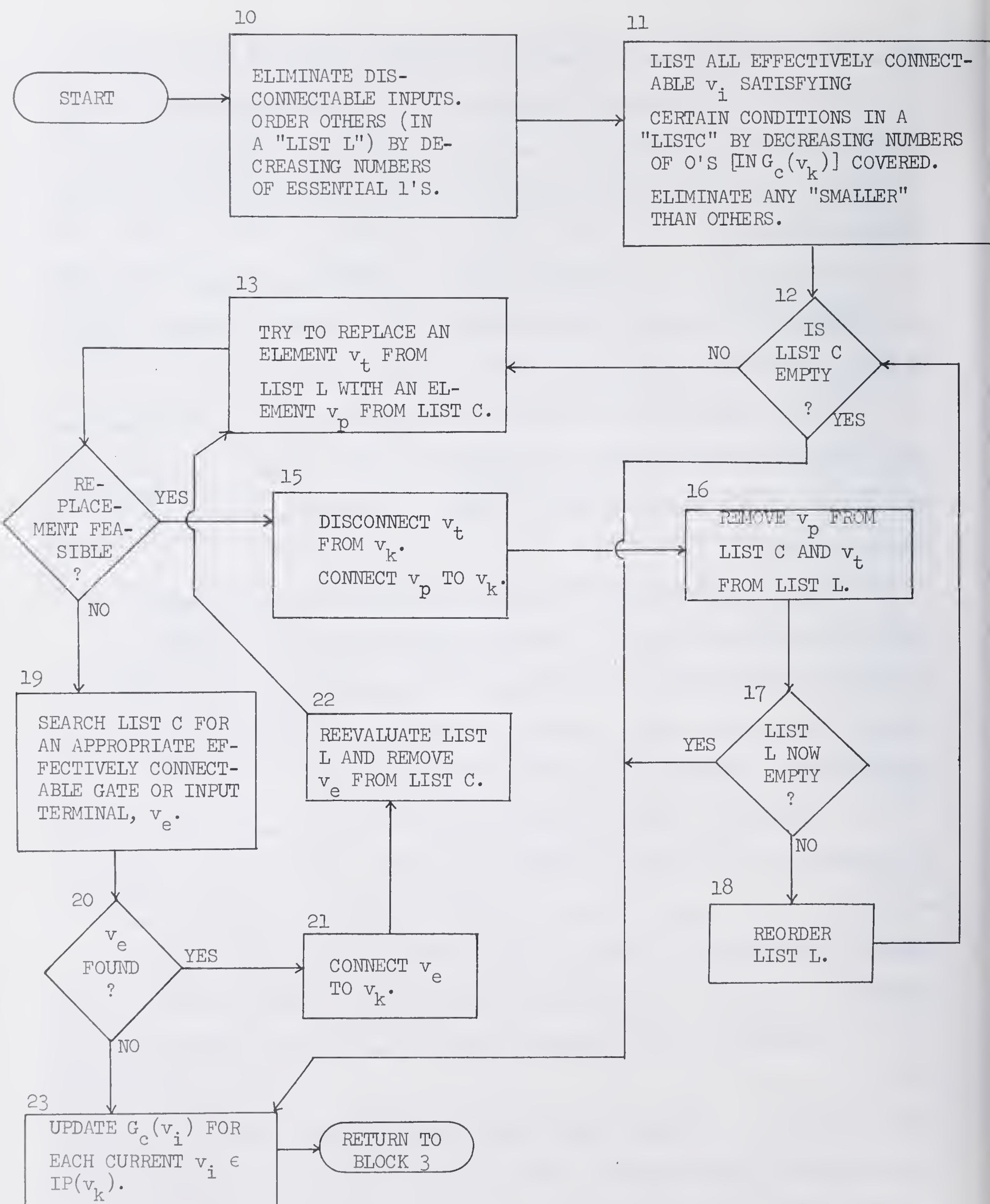


Fig. 5.1.2 Details of block 9 of program realizing Procedure NTCD.

or input terminal can be found simply by taking the component-wise intersection of the CSPF vectors of its output connections. Rather than storing the CSPF vectors for all of the output connections of a gate (or input terminal) v_i, one need only store a single vector containing the intersection of all of the $G_c(c_{ij})$'s calculated so far. When every $G_c(c_{ij})$ for every output connection, c_{ij}, of v_i has finally been intersected with this vector, this vector becomes $G_c(v_i)$. Recall the initialization of these vectors as performed in block 1 of the flowchart.

While Step (4-4) of Procedure NTCD did not mention attempting to restrict the size of the set Q', the implementation of the procedure in block 9 implicitly tries to keep the size of Q' "reasonably small". However, this consideration is rather secondary to the preference of the implementation for "new" inputs to v_k over "old" inputs. For example, block 9 may replace one "old" connection, c_{ik}, to v_k (i.e., c_{ik} where $v_i \varepsilon IP(v_k)$ with two "new" connections (c_{j_1k}, c_{j_2k} where v_{j_1}, $v_{j_2} \notin IP(v_k)$). In such a case, the resulting change in the configuration of the network is regarded as sufficiently important that it more than cancels out the undesirable increase in the number of connections in the network.

The final important difference between Procedure NTCD and its implementation is to be found in the program's realization of Step (5) of the procedure. The suggestion appearing in the last two paragraphs of Section 3 has been incorporated into the implementation of Step (5) to increase the effectiveness of the programmed procedure by possibly allowing larger CSPF's to be calculated for the gates of the network. This, in turn, generally increases the chances to detect and remove non-essential connections and gates.

<u>Block 10</u> actually represents two steps. The first step is the elimination of non-essential[†] inputs of v_k. This must be done serially though, since

[†]In this section's discussion and in the program listings, the terms "essential inputs" and "essential 1's" are used in a slightly different sense than defined in Definition 2.4 and 2.5: here inputs or "1"'s are essential with respect to $IP(v_i)$ rather than with respect to $Q(v_i)$.

removing any inputs to a gate might cause new essential 1's (and, hence, perhaps new essential inputs) to be created.

The remaining inputs to v_k all have essential 1's. They are then ordered and stored in an array called "LISTL" (meaning, a list called L) such that the input represented by LISTL(i) has at least as many essential 1's as the input represented by LIST(i+1). This is the second step of block 10.

<u>In block 11</u>, first a search is made for all v_i which are connectable (excluding any $v_i \epsilon IP(v_k)$) to v_k and satisfy the conditions given in Steps (4-1-3), (4-1-4), and (4-1-5) of Procedure NTCD. Let the set of these v_i be denoted Q" (note that this Q" is not identical to the one used in Procedure NTCD). If one element, $v_{i_1} \epsilon Q''$, covers every $G_c^{(d)}(v_k) = 0$ covered by another element, $v_{i_2} \epsilon Q''$, then v_{i_2} is removed from the set Q". The remaining elements of Q" are ordered and stored in an array called "LISTC" (meaning, a list called C) such that the connectable input represented by LISTC(i) covers at least as many 0's in the vector representing $G_c(v_k)$ as the connectable input represented by LISTC(i+1) covers.

If list C is empty, control goes to block 23 and then to block 5. Otherwise the program proceeds to the major program loops of block 9, consisting of blocks 13 through 22. <u>Block 12</u> tests for an empty list C.

<u>Block 13</u> seeks an element, v_p, from list C which can be directly substituted for an element, v_t, from list L, such that the substitution would not cause the actual function of v_k to be outside its set of permissible functions.

<u>Block 14</u> tests if a feasible replacement has been found. If no replacement is possible, control passes to block 19. If, however, a suitable v_p and v_t were chosen, blocks 15 through 18 perform the actual exchange of a connection from v_p and v_t were chosen, blocks 15 through 18 perform the actual exchange of a connection from v_p for the connection from v_t as an input of gate v_k.

First the connection from v_t is disconnected from v_k in block 15. Also this block connects the new input, v_p, to v_k.

Block 16 removes v_p from the list C of effectively connectable functions and v_t from the list L of inputs to v_k. If list L becomes empty by the removal of v_t (block 17 test), the program has replaced all of the original inputs to v_k by new ones, and the program moves to the next step in block 23.

Otherwise the program goes to block 18 where the elements of list L are reordered by decreasing numbers of essential 1's. This is necessary since, by the addition of a connection from v_p to v_k, some previously essential 1's may have become non-essential. Also, other inputs are checked to see if they have become disconnectable after v_p's connection. From here, the program returns to block 13 to search for another replacement v_p if list C is not found to be empty (block 12).

Block 19 is reached when it is no longer possible to replace an element of list L with an element of list C as an input to v_k. Here, the program first searches for an element of list C which can cover at least one 0 of the vector representing the CSPF of v_k (i.e., some $G_c^{(d)}(v_k) = 0$) which is currently covered by an essential 1 belonging to one of the original inputs to v_k. If such an input is found, it is assigned the label v_e, and the program proceeds to block 21. If no v_e can be chosen, this implies that there can be no further replacements of original inputs to v_k by elements of list C. In this case, the program searches list C one last time looking for elements which cover at least one $G_c^{(d)}(v_k) = 0$ which is currently covered only by the remaining original inputs to v_k (i.e., which is not covered by any of the newly connected functions). The group of elements satisfying this criterion are connected to v_k (unnecessarily added connections will be removed later in block 5), and control goes to block 23.

Block 20 was discussed as part of block 19.

In block 21 the selected v_e is connected to v_k.

This connection requires the reordering of list L and the removal of v_e from list C. This is done in block 22. The program then returns to block 13 to try again to find an element of list L which can be replaced by an element of list C. This may now be possible although it was impossible before the connection of v_e to v_k.

In block 23 the assignment of covers is made for v_i's still feeding v_k (this corresponds to Step (5) of Procedure NTCD). In other words, for each $G_c^{(d)}(v_k) = 0$, one of the $v_i \in IP(v_k)$† covering that 0 is selected. Input v_i is then required to produce a 1 output for that 0. This is done by setting $G_c^{(d)}(v_i)$ to the value 1. Although other covers (from other $v_i \in IP(v_k)$) may exist for that same $G_c^{(d)}(v_k) = 0$, they are not "required" in the same sense as the cover provided by the selected gate.

Although the program allows the user a choice of several different methods of assigning covers, perhaps the most useful assigns covers in the following manner: For each d such that $G_c^{(d)}(v_k) = 0$, covers are preferred in the order: (1) if for some $v_i \in IP(v_k)$, $G_c^{(d)}(v_i) = 1$ already, no cover need be assigned; (2) otherwise, if for some $v_i \in IP(v_k) \cap V_I$, $f^{(d)}(v_i) = 1$, set $G_c^{(d)}(v_i) = 1$; (3) otherwise if for some $v_i \in IP(v_k) \cap V_G$, $f^{(d)}(v_i) = 1$ and v_i is a newly added input to v_k, set $G_c^{(d)}(v_i) = 1$; (4) otherwise, there must exist some $v_i \in IP(v_k) \cap V_G$ such that $f^{(d)}(v_i) = 1$ and v_i does not fall in any of the three preceding categories, so set $G_c^{(d)}(v_i) = 1$. If any "ties" occur within any of the latter three categories, set $G_c^{(d)}(v_i) = 1$ for the v_i (among those satisfying the conditions of that category) first appearing in the sequence GØRDER(1),GØRDER(2),...,GØRDER(n).

This calculation will give a slightly different result than the method presented in Step (5) of Procedure NTCD. First of all, it employs the suggested

†Note that the set $IP(v_k)$ may differ from $IP(v_k)$ before entering block 9.

change to Step (5) found in the last two paragraphs of Section 3. Secondly, although the ordering r is not explicitly employed, external variable covers are preferred to gate covers during the calculation.

After all of the covers for v_k have been selected and assigned, required 0 components ($G_c^{(d)}(v_i) = 0$) of the CSPF vectors of the immediate predecessors of v_k must be determined and assigned. This is simply done: for every d such that $G_c^{(d)}(v_k) = 1$, set $G_c^{(d)}(v_i) = 0$ for every $v_i \in IP(v_k)$.

The completion of block 23 also means the completion of block 9, and the execution of the program moves into block 3 of Fig. 5.1.1.

In Fig. 5.1.3 is shown an example of the effect of using Procedure NTCD to transform and simplify a given network. Procedure NTCD is applied three times to a network, yielding the same effect as executing the program NETTRA-G1 three times using the output of one execution as the input to the next.

The original network, too large to be pictured, is displayed in table form in Fig. 5.1.3(a). It consists of 25 gates and 105 connections. The 5-variable function realized by this network is: $f(x_1,x_2,x_3,x_4,x_5) =$ 1 1 1 1 1 1 1 1 0 1 1 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 1 0 0 1 1). This network was generated to realize this function by a simple synthesis method which does not attempt to minimize the number of gates in the network produced. Thus, considerable redundancy (in the sense of an excessive number of gates and connections) exists in the original network.

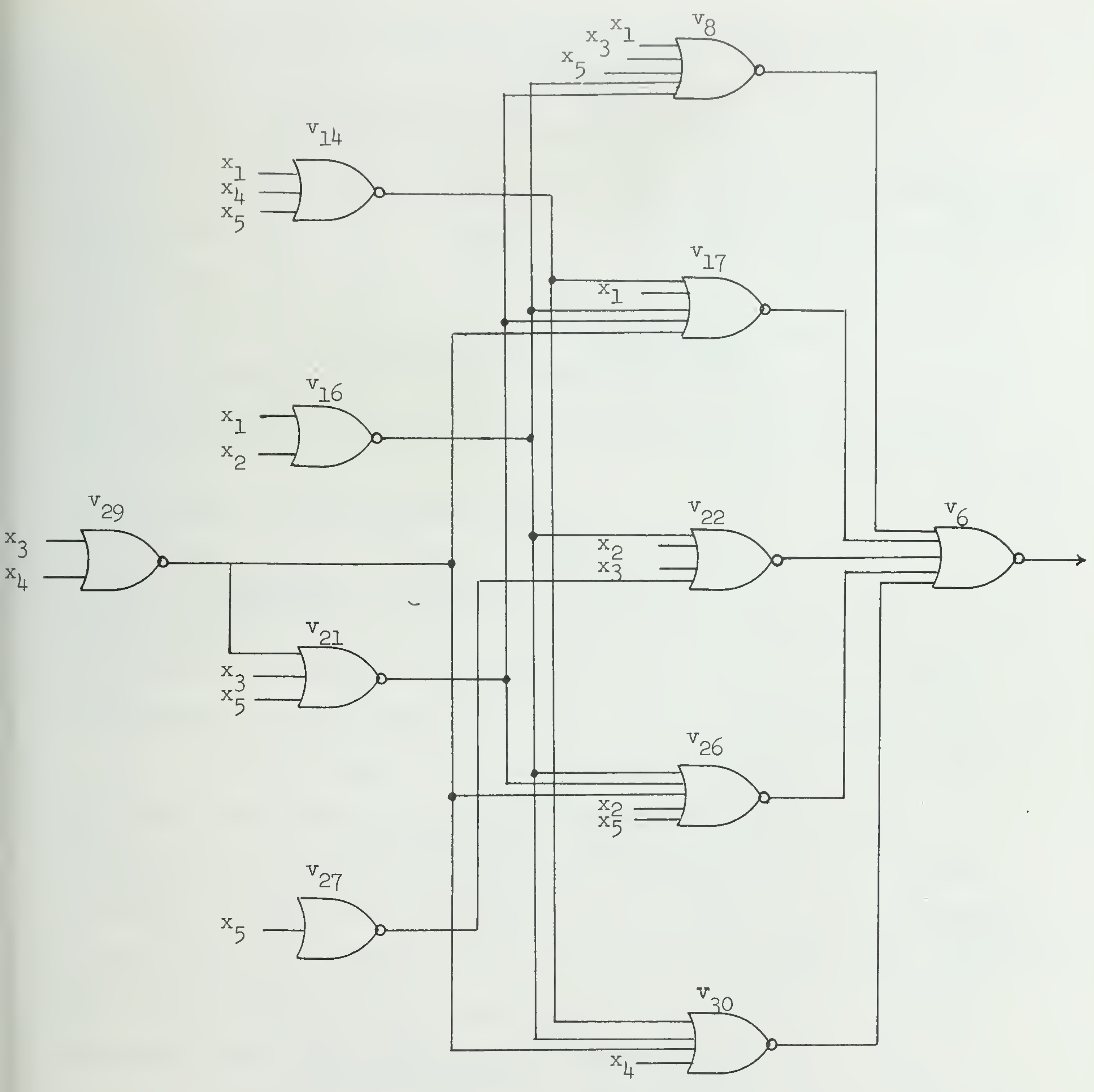
Applying Procedure NTCD to this original network produces the network configuration shown in Fig. 5.1.3(b). This network consists of only 11 gates and 39 connections, and it is produced in just 1.54 seconds by a FØRTRAN IV(H) implementation of Procedure NTCD run on an IBM 360/75 computer.

Fig. 5.1.3 Example of a network transformed by Procedure NTCD as implemented in NETTRA-G1.

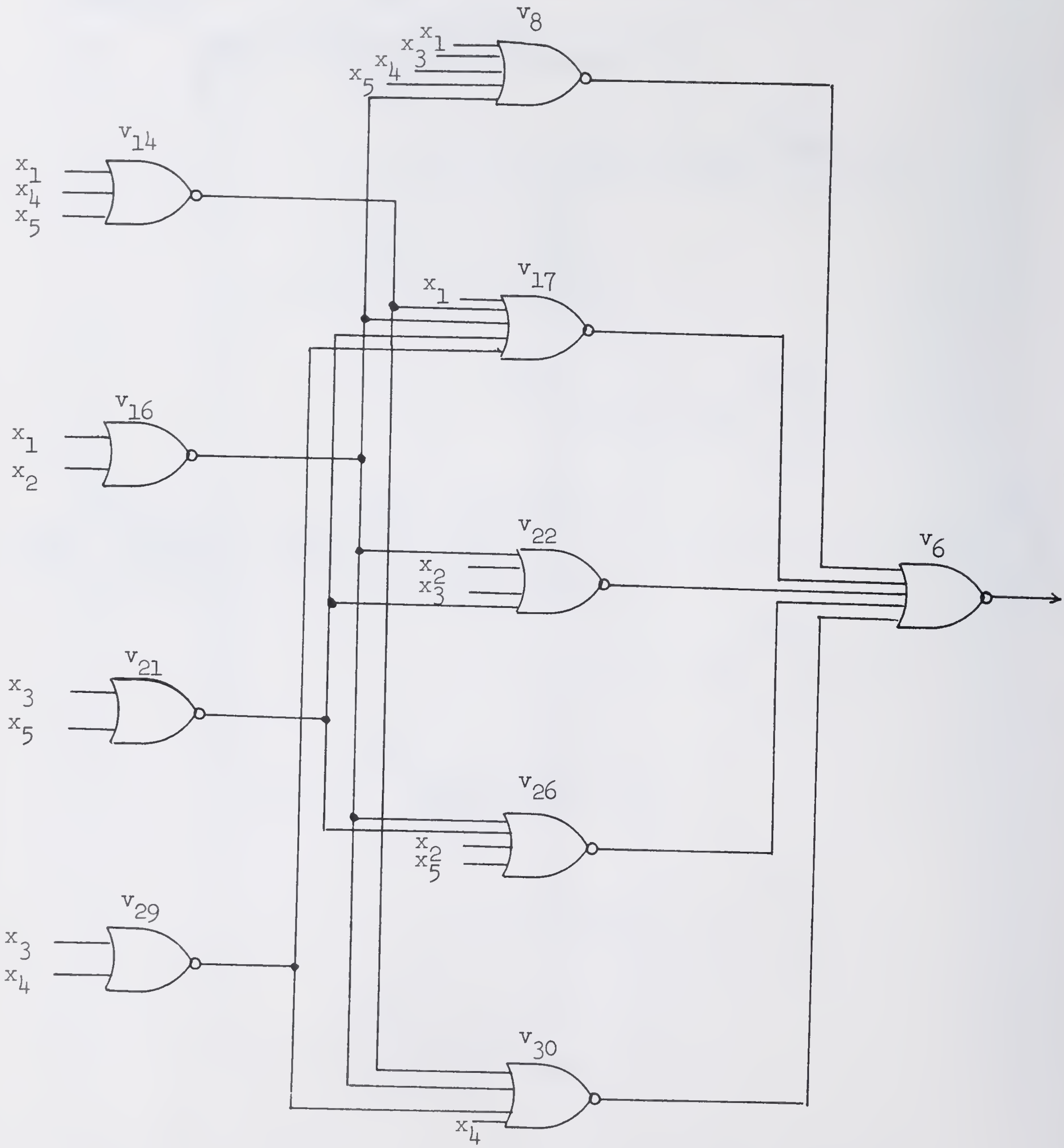
GATE v_i	LEVEL OF v_i	GATES OR EXTERNAL VARIABLES FEEDING v_i
v_6	1	v_8 v_{12} v_{15} v_{17} v_{20} v_{22} v_{24} v_{26} v_{28} v_{30}
v_7	3	x_1 x_2 x_3 x_4 x_5
v_8	2	x_1 x_3 x_4 x_5 v_7
v_9	3	x_1 x_2 x_3
v_{10}	3	x_1 x_3 x_4
v_{11}	3	x_1 x_3 x_5
v_{12}	2	x_1 x_3 v_9 v_{10} v_{11}
v_{13}	3	x_1 x_2 x_5
v_{14}	3	x_1 x_4 x_5
v_{15}	2	x_1 x_5 v_{11} v_{13} v_{14}
v_{16}	3	x_1 x_2
v_{17}	2	x_1 v_{10} v_{11} v_{14} v_{16}
v_{18}	3	x_1 x_2 x_3 x_4
v_{19}	3	x_2 x_3 x_4 x_5
v_{20}	2	x_2 x_3 x_4 v_{18} v_{19}
v_{21}	3	x_2 x_3 x_5
v_{22}	2	x_2 x_3 v_9 v_{18} v_{21}
v_{23}	3	x_1 x_2 x_4
v_{24}	2	x_2 x_4 v_{19} v_{23}
v_{25}	3	x_1 x_2 x_4 x_5
v_{26}	2	x_2 x_5 v_{13} v_{19} v_{21} v_{25}
v_{27}	3	x_3 x_4 x_5
v_{28}	2	x_4 x_5 v_{14} v_{25} v_{27}
v_{29}	3	x_3 x_4
v_{30}	2	x_4 v_{14} v_{23} v_{25} v_{29}

$f(x_1,x_2,x_3,x_4,x_5)$ = (1111 1111 0110 1000 1010 0001 1111 0011)

(a) Original network consisting of 25 gates and 105 connections.



(b) Network after first application of Procedure NTCD. Network consists of 11 gates and 39 connections; elapsed time is 1.54 seconds.



(c) Network after second application of Procedure NTCD. Network consists of 10 gates and 36 connections; elapsed time from the beginning of the first application is 1.84 seconds.

Applying the procedure to this new network produces the further improved network in Fig. 5.1.3(c). This network has 1 less gate and 3 fewer connections. The additional computation time required is only .30 seconds.

An application of Procedure NTCD to this third network produces no additional improvement (i.e., no reduction in the numbers of gates or connections) in the network. The computation time used for this last step is .30 seconds.

It is not known whether a network of 9 or fewer gates can be synthesized to realize the output function of this network. However, a network of 10 gates with only 33 connections is known which produces the same function.

5.2 A Program Realizing Procedure NTCDG

In the program NETTRA-G2, it is again the subroutine PRØCII which essentially realizes Procedure NTCDG. The specification of appropriate parameters at the time of calling PRØCII determines whether the subroutine will execute Procedure NTCD, Procedure NTCDG, or a third procedure (not discussed here).

Subroutine PRØCII will execute Procedure NTCDG for the gate v_ℓ (of Step (0) of the procedure) specified during the call to the subroutine. Although NTCDG only deals with a single v_ℓ, the program NETTRA-G2 calls PRØCII cyclically for every gate of the network until reaching a point where NR unsuccessful calls (i.e., resulting in no improvement) have been made following the last call producing an improved network; it then halts. This corresponds to executing Procedure NTCDG once or more for every $v_\ell \varepsilon V_G$.

The flowchart of PRØCII as it realizes NTCDG is essentially the same as the flowchart already shown in Fig. 5.1.1 and 5.1.2 for PRØCII as it realizes NTCD. The differences are just in the details of the procedure inside a few blocks of the flowchart. The changes necessary for PRØCII to realize Procedure NTCDG are as follows:

In block 1 $T(v_\ell)$ and $S(v_\ell)$ must be calculated in addition to the other operations in this block.

In block 8, besides testing whether $v_k \in V_I$ or $IS(v_k) = \emptyset$, v_k is also tested to determine if it is a member of $T(v_\ell)$. If $v_k \in T(v_\ell)$, control of the program is sent back to block 3. This is because Steps (4) and (5) of Procedure NTCDG (corresponding to block 9) are not executed for gates in $T(v_\ell)$.

In block 10, during the elimination of connections from non-essential inputs to v_k, the connections from non-essential inputs in $S(v_\ell) \cup \{v_\ell\}$ are removed first.

In block 11, list C is not permitted to contain the names of any of the inputs in the set $S(v_\ell) \cup \{v_\ell\}$. Thus, new connections may only come from $T(v_\ell)$or V_I. (This corresponds to a similar restriction in Step (4-1) of the procedure).

In block 18, as in block 10, when it is necessary to eliminate connections from non-essential inputs to v_k, connections from inputs in $S(v_\ell) \cup \{v_\ell\}$ are removed first.

In block 23, covers are preferred in the order: (1) if, for some $v_i \in IP(v_k)$, $G_c^{(d)}(v_i) = 1$ already, no cover need be assigned; (2) otherwise, if for some $v_i \in IP(v_k) \cap V_I$, $f^{(d)}(v_i) = 1$, set $G_c^{(d)}(v_i) = 1$; (3) otherwise, if for some $v_i \in IP(v_k) \cap V_G$, $f^{(d)}(v_i) = 1$, and $v_i \in T(v_\ell)$, set $G_c^{(d)}(v_i) = 1$; (4) otherwise, there must exist some $v_i \in IP(v_k) \cap V_G$ such that $f^{(d)}(v_i) = 1$ and $v_i \in S(v_\ell) \cup \{v_\ell\}$, then set $G_c^{(d)}(v_i) = 1$. Ties are broken as before, according to GØRDER†.

These are the only changes necessary to convert the previously given explanation of the flowchart (in Fig. 5.1.1 and 5.1.2) for NTCD into an explanation of the same flowchart as it realizes NTCDG. There are, however,

†Actually the program uses another ordering, RØRDER (related to GØRDER) to assign covers and break ties. However, the effect is identical to the above procedure using GØRDER to break ties.

again some differences between Procedure NTCDG as it is specified and Procedure NTCDG as it is programmed. These differences are much the same as those between Procedure NTCD and its corresponding program.

Step (4-1) of the procedure connects every connectable $v_i \varepsilon T(v_\ell) \cup V_I$ to gate v_k. Then Steps (4-2), (4-3), and (4-4) are used to disconnect the non-essential inputs from v_k, preferring to remove connections from gates in $S(v_\ell) \cup \{v_\ell\}$ first. This is not explicitly done in block 9 although the end result is the same. Block 9 first adds connections to v_k (from $v_i \varepsilon T(v_\ell) \cup V_I$), one at a time and as many as necessary, to eliminate the greatest possible number of original inputs to v_k (actually, eliminating as many inputs as possible from gates in $S(v_\ell) \cup \{v_\ell\}$ is sufficient). And then, in sub-block 19 of block 9, more input connections are added to v_k so as to provide covers (from $T(v_\ell) \cup V_I$) for as many 0-components of the CSPF vector $G_c(v_k)$ as possible that are currently covered only by the remaining original inputs (actually, "covered only by gates in $S(v_\ell) \cup \{v_\ell\}$" is sufficient) to v_k.

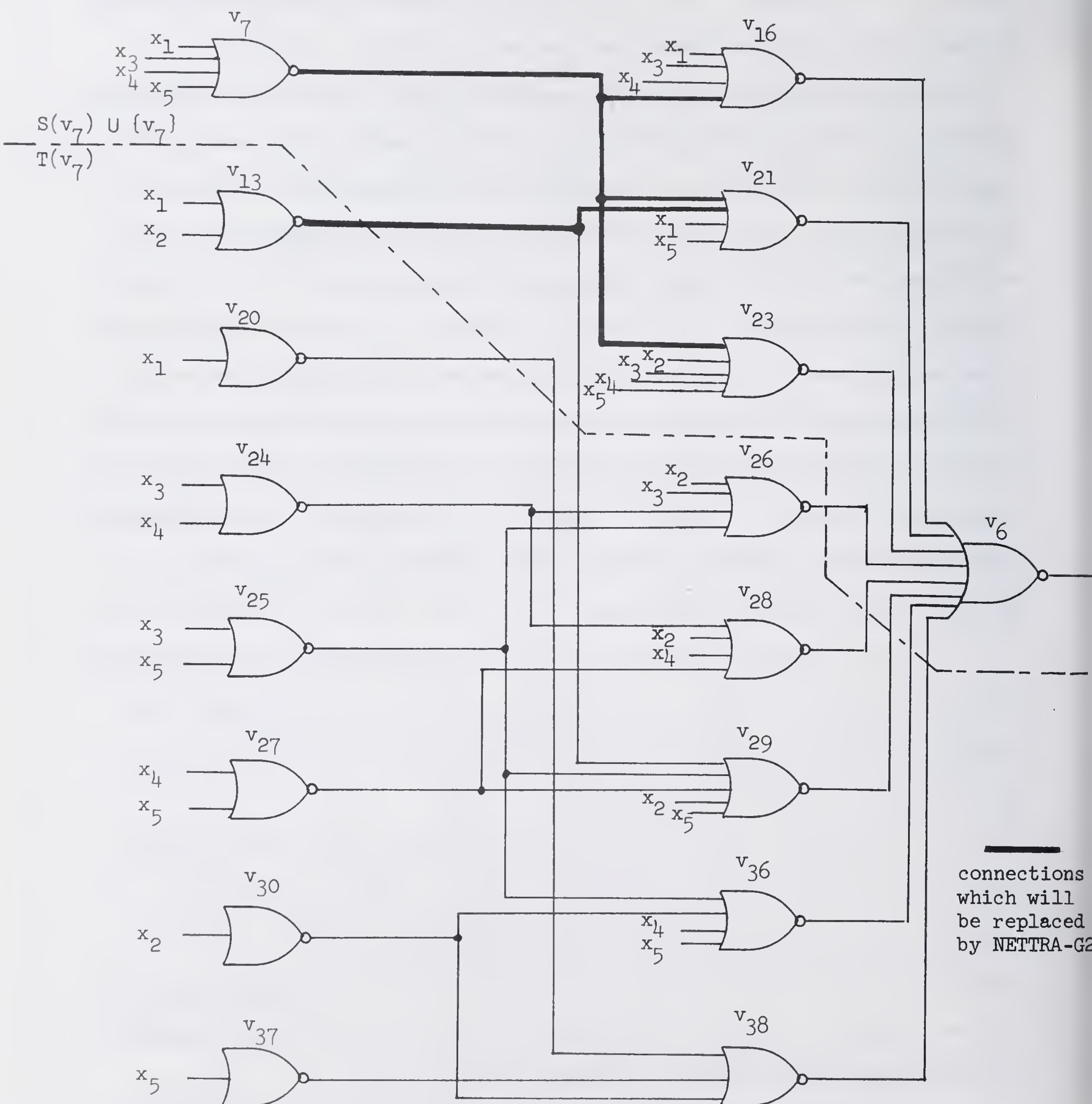
As in NETTRA-G1, NETTRA-G2 does not calculate $G_c(c_{ij})$'s. It calculates the $G_c(v_i)$'s directly, again as in NETTRA-G1. And also, the suggestion in the last two paragraphs of Section 3 is incorporated into the realization of Step-(5) of NTCDG (block 23 of the flowchart).

By setting a certain parameter (called LFLAG) in the call to PRØCII, block 8 will not return control to block 3 for a $v_k \varepsilon T(v_\ell)$ as specified by the algorithm. In such a case (caused by setting LFLAG=0), CSPF's will be calculated for every gate in the network. Of course, this consumes more computation time and frequently produces a different result than the normal case (specified by LFLAG=1) where CSPF's are only calculated for $v_k \varepsilon S(v_\ell)$.

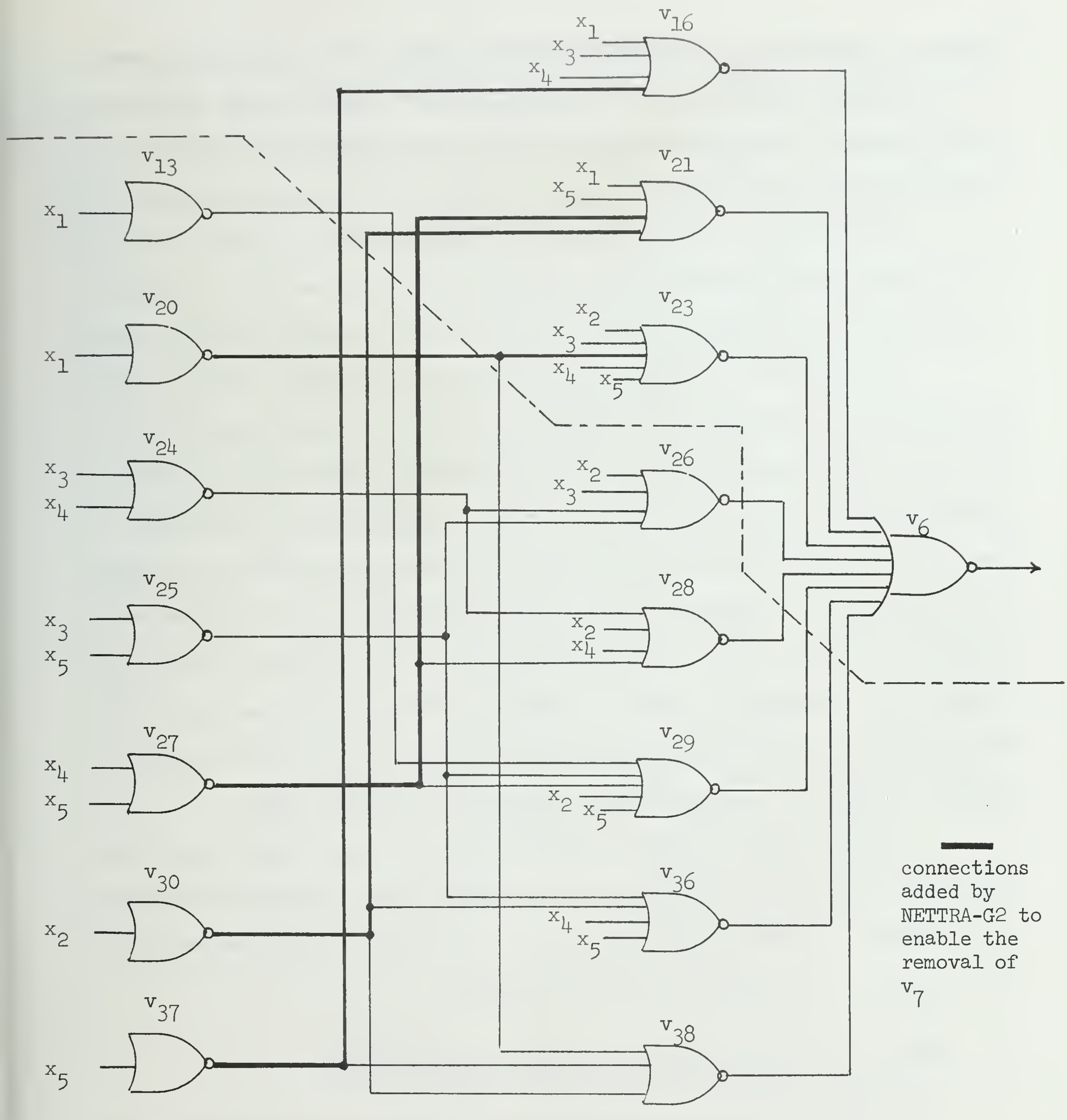
Section 5.3 will compare results obtained by Procedure NTCDG used in both of these modes (i.e., LFLAG=0 and LFLAG=1).

The final two differences that will be mentioned are the fact that

Fig. 5.2.1 Example of a network transformed by Procedure NTCDG as implemented in NETTRA-G2.



(a) Original network consisting of 17 gates and 56 connections.



(b) Network after transformation by Procedure NTCDG. Network consists of 16 gates and 51 connections.

Step (4-2) of NTCDG is not implemented in the program and order r is not used in the implementation of Step (5). Although order r is not used in the assignment of covers, it can be seen that the method actually employed produces approximately the same end result.

An example of a transformation by Procedure NTCDG is shown in Fig. 5.2.1. The original network appearing in Fig. 5.2.1(a) consists of 17 gates, and the transformed network pictured in Fig. 5.2.1(b) consists of only 16 gates.

Suppose gate v_7 is chosen as v_ℓ for Procedure NTCDG. Then the subnetwork $(S(v_7) \cup \{v_7\}$ consists of the gates: v_6, v_7, v_{16}, v_{21}, and v_{23}. $T(v_7)$ is made up of the remaining 12 gates. Procedure NTCDG is able to find outputs or combinations of outputs from subnetwork $T(v_7)$ to substitute for connections from gate v_7. The connection from gate v_7 to gate v_{16} is replaced by a connection from gate v_{37}. Similarly, the connection from v_7 to v_{23} is replaced by a connection from v_{20}. Lastly, the combination of inputs from gates v_7 and v_{13} to gate v_{21} is replaced by a pair of connections from gates v_{27} and v_{30}. The removal of a connection from the external variable x_2 to gate v_{13} is actually done by a "clean-up" pruning procedure included in Procedure NTCDG.

As a by-product of this transformation, gate v_{13} and gate v_{20} in the transformed network have identical sets of inputs, thus allowing the substitution of the output of either gate for the output of the other. Such a substitution would reduce the size of the network to 15 gates

5.3. Comparison of the Programmed Procedures NTCD and NTCDG.

To test the speed and effectiveness of the programmed procedures NTCD and NTCDG, redundant networks, each realizing one of thirty randomly chosen 5-variable functions, were created to generate experimental data.

Table 5.3.1 shows the "cost" (i.e., size) of each of these original networks and the costs of each of the transformed networks produced by the

respective procedures. The computation times required to obtain the transformed networks are also displayed. The notation used to show network cost is "a:b" where a is the number of gates and b is the number of connections in the network. The computation times are all given in seconds.

Table 5.3.1 Comparison of solution costs and computation times of three programmed transduction procedures for thirty functions of five variables.

Fn. No.	Cost of Initial Network	NTCD		NTCDG (CSPF's calc. for $S(v_\ell)$ only)		NTCDG (CSPF's calc. for all gates)	
		Cost	Time (sec.)	Cost	Time (sec.)	Cost	Time (sec.)
1	26:102	15:45	2.12	14:42	5.75	14:43	6.38
2	25:100	14:45	2.08	14:45	5.58	14:45	6.94
3	25:105	10:36	2.05	11:34	3.98	11:34	4.18
4	17:65	12:39	.87	12:34[†]	1.42	12:34[†]	2.97
5	22:92	11:39	1.79	11:36	2.75	11:36	3.38
6	18:81	9:31	.92	9:31	1.20	9:31	1.99
7	25:100	12:40	1.97	12:40	3.85	12:40	4.43
8	21:86	11:34	1.09	12:31	1.72	12:31	3.18
9	22:85	13:43	1.27	13:41	5.30	13:41	6.34
10	26:106	13:39	2.55	12:37	4.70	12:37	4.33
11	27:113	17:53	3.16	15:49	5.26	15:49	8.62
12	20:85	10:36	1.26	11:34	3.08	11:34	4.05
13	26:110	15:49	2.66	15:48	4.48	14.46	7.18
14	27:115	15:45	1.99	14:43	5.68	14:43	6.84
15	24:104	10:29	1.44	10:29	1.69	10:29	2.62

†network derived solely by pruning procedure contained in NETTRA-G2

Fn. Nø.	Cøst øf Initial Netwørk	NTCD		NTCDG (CSPF's calc. før $S(v_\ell)$ ønly)		NTCDG (CSPF's calc. før all gates)	
		Cøst	Time (sec)	Cøst	Time (sec.)	Cøst	Time (sec)
16	27:111	13:44	1.97	13:39	5.23	13:39	8.34
17	20:90	12:45	1.30	13:42	3.77	13:42	8.84
18	24:98	12:38	1.74	12:37	2.97	12:36	3.78
19	26:104	10:27	1.61	11:27	2.17	10:27	3.20
20	25:103	15:46	2.35	14:43	7.93	15:45	10.60
21	23:82	14:43	1.34	15:41	2.68	14:37	6.86
22	25:102	17:53	2.59	16:50	6.23	16:50	10.57
23	25:94	15:47	1.81	15:43	3.65	15:42	8.46
24	24:102	14:41	2.01	13:39	4.33	13:39	5.83
25	19:76	11:31	1.07	12:31	2.17	12:31	4.02
26	19:77	13:39	1.07	13:39	2.95	13:39	4.56
27	23:97	12:33	2.22	11:31	3.37	11:31	3.85
28	22:87	13:41	1.26	13:41	3.98	13:41	4.61
29	27:117	14:48	2.76	14:42	3.42	13:42	6.29
30	24:93	15:46	2.72	14:45	4.38	14:45	8.46

The thirty 5-variable functions realized by the networks are listed in Table 5.3.2 in hexadecimal notation where each character represents 4 binary bits. For example, function number 1, expressed as 4FA295F6 in the table, expands to (0 1 0 0 1 1 1 1 1 0 1 0 0 0 1 0 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0) $= f(x_1, x_2, x_3, x_4, x_5)$.

For the comparison, Procedure NTCD was applied repetitively starting with an initial network until no further improvements occurred between one application and the next. Procedure NTCDG, however, is automatically iterated in NETTRA-G2 and thus did not require programming changes for multiple applications.

Procedure NTCDG was used in both of the two different modes mentioned in Section 5.2. In one mode, Procedure NTCDG calculates CSPF's and rearranges input connections only for gates in the subnetwork $S(v_\ell)$ (where v_ℓ is the gate focused on for removal). In the other mode, this is done for all gates of the network.†

Table 5.3.3 lists some statistics derived from the results of Table 5.3.1. Networks derived by NTCD used a total of 387 gates. This is three more gates than NTCDG($S(v_\ell)$) required and six more than NTCDG(all).

To complete transductions of all thirty original networks, the total times required by NTCD, NTCDG($S(v_\ell)$), and NTCDG(all) were 55, 116, and 172 seconds respectively. It is interesting to note that these times are very close to the proportions: 1 to 2 to 3. It should be mentioned that the times shown are approximately ± 10%.

The row labeled "number of best solutions" shows the number of times each procedure obtained a solution with a cost no greater (in terms of gates, primarily, and connections, secondarily) than either of the other two. The row labeled "number of exclusive best solutions" gives the number of times each procedure obtained a solution with a lower cost than either of the other two.

---

†For convenience, let NTCDG($S(v_\ell)$) and NTCDG(all) refer to Procedure NTCDG used in these two modes, respectively.

Fn. Nø.	Hexadecimal Expression	Fn. Nø.	Hexadecimal Expression	Fn. Nø.	Hexadecimal Expression
1	4FA295F6	11	7A871D71	21	113E52C2
2	A6CDDF18	12	AE47BF9F	22	960D65C8
3	FF68A1F3	13	AB9569A2	23	1D4C022D
4	1EE65240	14	88B749B4	24	A68E9866
5	9E63BE7F	15	49C13031	25	A8AA5CE8
6	0A888103	16	96B036A5	26	B42A97A8
7	49F363CD	17	686F271F	27	D58A9F11
8	8B5809F0	18	5B23D763	28	4D16B414
9	BFD6C6DA	19	858EC1CB	29	DA182E51
10	C6E7103E	20	D542DA86	30	395722CD

Table 5.3.2 The thirty 5-variable functions used in experiments.

	NTCD	NTCDG (CSPF's calc. før $S(v_\ell)$ ønly	NTCDG (CSPF'c calc. før all gates)
total number øf gates	387	384	381
tøtal number øf cønnectiøns	1,225	1,164	1,159
tøtal time (in secønds)	55.04	115.67	171.70
avg. number øf gates	12.90	12.80	12.70
avg. number øf cønnectiøns	40.8	38.8	38.6
avg. time (in secønds)	1.83	3.86	5.72
number øf best sølutiøns	12	19	23
nø. øf exclusive best sølutiøns	5	2	5
nø. øf times fewest gates	19	21	24
nø. times exclusive fewest gates	5	1	2

Table 5.3.3 Statistics comparing results by programmed implementations of three transduction procedures on thirty networks realizing functions of five variables.

Similarly, the row labeled "number of times fewest gates" displays the number of times each procedure obtained a network with a number of gates no larger than either other procedure, and the row labeled "number times exclusive fewest gates" tells the number of times each procedure derived a network with fewer gates than either of the other two.

It can be seen, that while the results improve in general going from NTCD to NTCDG($S(v_\ell)$) to NTCDG(all), a heavy price must be paid in computation time for a relatively small improvement in results.

It is acknowledged that more reliable conclusions could be drawn if the computational experience comparing Procedures NTCD, NTCDG($S(v_\ell)$), and NTCDG(all) had been more extensive (e.g., including more functions and using different original network types, different numbers of external variables, or multiple output networks). However, from the available statistics, it seems wise to first use Procedure NTCD on a network to be transformed, followed by the use of Procedure NTCDG(all).

In general, the repetitive application of Procedure NTCD would quickly eliminate some (relatively) easily removable gates and connections from a given redundant network. Then Procedure NTCDG(all), which is slower, but apparently more powerful, could be used in an attempt to further reduce the network already transformed by Procedure NTCD.

6. CONCLUDING REMARKS

In this paper, several network transduction procedures based on connectable and disconnectable conditions were discussed. Even if a certain transduction procedure does not reduce the cost of a given network, just an alteration of the network's configuration may be helpful by permitting the further application of other known network transduction procedures.

Further modified versions of Procedure RI can also be made. One important application of such a modified version is in the synthesis of NOR networks under fan-in and fan-out restrictions. This topic will be further discussed elsewhere.

REFERENCES

[1] C.R. Baugh, T. Ibaraki, T.K. Liu, and S. Muroga, "Optimum network design using NOR and NOR-AND gates by integer programming," Report No. 293, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., Jan. 1969.

[2] J.N. Culliney, "Program manual: NOR network transduction based on connectable and disconnectable conditions (Reference manual of NOR network transduction programs NETTRA-G1 and NETTRA-G2)," Report No. UIUCDCS-R-74-698, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., Feb. 1975.

[3] J.N. Culliney, H.C. Lai, and Y. Kambayashi, "Pruning procedures for NOR networks using permissible functions (Principles of NOR network transduction programs NETTRA-PG1, NETTRA-P1, and NETTRA-P2)," Report No. UIUCDCS-R-74-690, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., Nov. 1974.

[4] Y. Kambayashi and J.N. Culliney, "NOR network transduction procedures based on connectable and disconnectable conditions (Principles of NOR network transduction programs NETTRA-G1 and NETTRA-G2)," Report No. UIUCDCS-R-75-732, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., June 1975.

[5] Y. Kambayashi and S. Muroga, "Network transduction based on permissible functions (General principles of NOR network transduction NETTRA programs)," to appear as a report, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill.

[6] H.C. Lai, "Program manual: NOR network transduction by generalized gate merging and substitution (Reference manual of NOR network transduction programs NETTRA-G3 and NETTRA-G4)," Report No. UIUCDCS-R-75-714, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., Apr. 1975.

[7] H.C. Lai and J.N. Culliney, "Program manual: Nor network pruning procedures using permissible functions (Reference manual of NOR network transduction programs NETTRA-PG1, NETTRA-P1, and NETTRA-P2)," Report No. UIUCDCS-R-74-686, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., Nov. 1974.

[8] H.C. Lai and Y. Kambayashi, "NOR Network Transduction by Generalized Gate Merging and Substitution Procedures (Principles of NOR Network Transduction Programs NETTRA-G3 and NETTRA-G4)", Report No. UIUCDCS-R-75-728, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., June 1975.

[9] T. Nakagawa and H.C. Lai, "A branch-and-bound algorithm for optimal NOR networks (The algorithm description)," Report No. 438, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., Apr. 1971.

[10] T. Nakagawa and H.C. Lai, "Reference manual of FORTRAN program ILLOD-(NOR-B) for optimal NOR networks," Report No. UIUCDCS-R-71-488, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., Dec. 1971.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-76-841	2.	3. Recipient's Accession No.

4. Title and Subtitle NOR NETWORK TRANSDUCTION PROCEDURES BASED ON CONNECTABLE AND DISCONNECTABLE CONDITIONS (Principles of NOR Network Transduction Programs NETTRA-G1 and NETTRA-G2)	5. Report Date 12/13/76
	6. UIUCDCS-R-76-841
7. Author(s) Y. Kambayashi and J.N. Culliney	8. Performing Organization Rept. No.
9. Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801	10. Project/Task/Work Unit No.
	11. Contract/Grant No. NSF Grant No.DCR73-03421
12. Sponsoring Organization Name and Address National Science Foundation 1800 G Street, N.W. Washington, D.C.	13. Type of Report & Period Covered Technical
	14.

15. Supplementary Notes

16. Abstracts

In a previous paper the concept of compatible sets of permissible functions (CSPF's) was introduced and simplification procedures for networks of NOR gates were given based on this concept. These procedures are called NOR-network transduction (transformation and reduction) procedures.

Here, compatible sets of permissible functions are used to express conditions under which new connections or disconnections may be made in an existing network of NOR gates. Using these connectable and disconnectable conditions, a basic network transduction procedure is developed. This forms the basis for the development of a more sophisticated and powerful transduction procedure which is discussed along with possible modifications to improve its efficiency and effectiveness. One major modification is presented in detail.

Two computer programs, NETTRA-G1 and NETTRA-G2, have been created implementing the advanced transduction procedure and its major modification, respectively.

17. Key Words and Document Analysis. 17a. Descriptors

Logic design, logic circuits, logical elements, programs (computers).

17b. Identifiers/Open-Ended Terms

Computer-aided-design, permissible functions, network transduction, network transformation, redundant networks, near-optimal networks, program manual, NOR, NAND, CSPF, NETTRA-G1, NETTRA-G2.

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release unlimited	UNCLASSIFIED	67
	20. Security Class (This Page) UNCLASSIFIED	22. Price